Question Answering System based on Tourism Knowledge Graph

Student Name : Manoli Christina

University of Macedonia

A thesis submitted for the degree of:
BACHELOR OF SCIENCE (BSc) IN APPLIED INFORMATICS

Thesis Committee:
Georgia Koloniari
Georgios Evangelidis
Christos Georgiadis

March 2022 – January 2023
Thessaloniki – Greece

Acknowledgments

I would like to acknowledge that this thesis is brought to you after months of research and constant effort to achieve the best possible outcome and it wouldn't be possible without the support and guidance of my supervisor, Georgia Koloniari. Therefore, I would like to, first of all, thank my supervisor who helped me achieve this result by providing a “rich” research environment and a set of interesting questions as well as suggestions without which this work wouldn't be where it is today.

Furthermore, I would like to thank the truly exceptional people in my life that always supported and motivated me throughout this whole process. My family. My friends. Thank you.

Lastly, I need to thank all the professors that throughout my academic career truly helped me develop my academic knowledge. Especially, I must thank my professor, Markos Zampoglou, who introduced me to Machine Learning and Neural Networks and imparted me with his knowledge about said topics, thus helping me create the passion and love that I possess today.

Περίληψη

Στη σημερινή εποχή, η κοινωνία μας βρίσκεται εν μέσω μιας μεγάλης αναπτυξιακής περιόδου, που περιλαμβάνει τη σημαντική έρευνα που διεξάγεται πάνω σε σημαντικές και χρήσιμες τεχνολογίες. Μεταξύ αυτών των τεχνολογιών είναι και οι Γράφοι Γνώσης (Knowledge Graphs) καθως και τα Συστήματα Απαντήσεων Ερωτήσεων (Question Answering Systems). Στην παρούσα εργασία, διεξάγουμε μια ενδελεχή και εκτενή έρευνα τόσο για τους Γράφους Γνώσης όσο και για τα Συστήματα Απαντήσεων Ερωτήσεων, προκειμένου να αναδείξουμε τις ικανότητες τους και τελικά να αναπτύξουμε το δικό μας Σύστημα Απαντήσεων Ερωτήσεων που χρησιμοποιεί τον πρωτότυπο Γράφο Γνώσης για τον Τουρισμό. Η μελέτη αυτή αποτελείται κυρίως από δύο μέρη. Πρώτον, αναλύουμε το θεωρητικό υπόβαθρο προκειμένου να κατανοήσουμε καλύτερα τις τεχνολογίες και να ορίσουμε τους θεμελιώδεις όρους και τα εργαλεία που θα χρησιμοποιήσουμε αργότερα στο δεύτερο πιο τεχνικό μέρος της πτυχιακής.

Πρώτα απ' όλα, συστήνουμε την έννοια των Γράφων Γνώσης – γράφοι διασυνδεδεμένων δεδομένων που αποτελούνται από οντότητες του πραγματικού κόσμου που μας ενδιαφέρουν, τις σχέσεις που τις συνδέουν μεταξύ τους και τα χαρακτηριστικά που βοηθούν στην καλύτερη περιγραφή τους. Έπειτα, σημειώνουμε τις γενικές πληροφοριες και στοιχεια που θα πρέπει να ξέρουμε για τους Γράφους Γνώσης – από το ιστορικό τους υπόβαθρο έως τη σημασία τους και τις διάφορες εφαρμογές τους. Επιπλέον, επεκτείνουμε αυτή τη βιβλιογραφική ανασκόπηση και στα συστήματα απάντησης ερωτήσεων. Επίσης, αναλύουμε την σχετική έρευνα που έχει διεξαχθεί γύρω από τον τομέα των Γνώσεων Τουρισμού.

Όσον αφορά το δεύτερο μέρος της παρούσας πτυχιακής, ξεκινάμε με την ανάλυση της διαδικασίας δημιουργίας του Τουριστικού Γράφου Γνώσης για τη μελέτη περίπτωσης της Σαντορίνης. Επιπλέον, παρουσιάζουμε λεπτομερώς τον τρόπο με τον οποίο δημιουργήσαμε το Σύστημα Απαντήσεων Ερωτήσεων αναπτύσσοντας ένα μοντέλο βασισμένο σε πρότυπα και εφαρμόζοντας τεχνολογίες όπως η Επεξεργασία Φυσικής Γλώσσας και τα state-of-the-art BERT Embeddings μεταξύ άλλων. Επιπλέον, κατά τη διάρκεια αυτής της ερευνητικής μελέτης πραγματοποιήσαμε πληθώρα διαφορετικών πειραμάτων προκειμένου να βελτιώσουμε την απόδοση του συστήματός μας.

Λέξεις Κλειδιά: *Γράφοι Γνώσης • Σύστημα Απαντήσεων Ερωτήσεων • Neo4j • Επεξεργασία Φυσικής Γλώσσας • BERT Embeddings • Μοντέλο βασισμένο σε πρότυπα • Έρευνα • Τουρισμός*

Χριστίνα Μανώλη

Ημερομηνία
26/01/2023

Abstract

Nowadays, our society finds itself in the midst of a great evolving period including the important research that is being conducted towards important and useful technologies. Amongst those technologies we can find **Knowledge Graphs (KG)** and **Question Answering Systems (QASs)**. In this paper, we conduct a thorough and extensive research on both Knowledge Graphs and Question Answering Systems in order to **highlight their potential** and ultimately **develop our own QA System** that **employs** our original, purpose-built (domain specific) **Knowledge Graph** for **Tourism**. This study mainly consists of **two parts**. Firstly, we analyze the **theoretical background** in order to better understand our technologies and set the fundamental terms and tools that we are going to later use on the second more **technical part** of the thesis.

First and foremost, we introduce the concept of Knowledge Graphs; *graphs of interconnected data which consist of **real-world entities of interest**, the **relations** that interconnect them and the **attributes** that help better describe them.* From there we move on to duly note the KG background information; from their **historical Background** to their **significance** and various **applications.** Moreover, we extend this literature review into Question Answering Systems as well. Additionally, we break down any **related research** that has been conducted around the field of **Tourism Knowledge Graphs**.

Regarding the second part of this thesis, we start by **analyzing the process of creating** our Tourism Knowledge Graph for the case study of Santorini. Furthermore, we present in detail how we built our Question Answering System by deploying a **Template-Based** model and implementing technologies like **Natural Language Processing** and state-of-the-art **BERT embeddings** amongst others. In addition, during this research study we have performed a plethora of different experiments in order to improve our systems performance.

Keywords: *Knowledge Graphs • Question Answering System • Neo4j • Natural Language Processing • BERT Embeddings • Template-based model • Research • Tourism*

Christina Manoli

Date
26/01/2023

Contents

List of Figures

List of Tables

List of Examples

List of Abbreviations

Abbreviation	Definition
KG	Knowledge Graph
TKG	Tourism Knowledge Graph
QA	Question Answering
QAS	Question Answering System
RDF	Resource Description Framework
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
IRI	Internationalized Resource Identifier
NLP	Natural Language Processing
W3C	World Wide Web Consortium
IE	Information Extraction
EL	Entity Linking
NER	Named Entity Recognition
RE	Relation Extraction
LPG	Labeled Property Graph
DKG	Domain-Specific Knowledge Graph
WTTC	World Travel and Tourism Council
GDP	Gross Domestic Product
YAGO	Yet Another Great Ontology
IR	Information Retrieval
TKGQA	Tourism Knowledge Graph Question Answering System
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network

Abbreviation	Definition
DMO	Destination Management Organization
GIS	Geographical Information System
KGQA	Knowledge Graph Question Answering system
SS	Sentence Similarity
LS	Linguistic Similarity
CRF	Conditional Random Field
RNN	Recurrent Neural Network
BiLSTM	Bidirectional Long Short-Term Memory
SVM	Support Vector Machine
GNN	Graph Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit

Chapter 1

Introduction

Nowadays, we can observe how the world and our lives face a gradual shift moving on from analog to incline towards a more digital nature. Furthermore, information is rapidly getting digitized and becoming available to a lot of individuals who strongly rely on it for carrying out their everyday tasks, searching for answers to their questions or gaining more knowledge. With that in mind, the big question is how do we handle and use that information/knowledge in the best possible way?

Knowledge Graphs can help us with that as it possesses the ability to integrate various heterogeneous information in order to create a "rich" and insightful semantic network capable of decision-making, deriving new information, recommendation, question answering as well as other applications. According to *H. Paulheim,* a knowledge graph can be defined as *a description of real-world entities and their interrelations, organized in a graph, able to define possible classes and relations of entities in a schema, allows for potentially interrelating arbitrary entities with each other and covers various topical domains* [44]. Knowledge Graphs (KGs), often referred to as semantic networks, are an emerging technology that have sparked an interest in recent years since the introduction of Google's Knowledge Graph in 2012. Various people in both research and business have realized its potential and have adopted it despite KGs being a technology at its "infancy".

Undoubtedly, humans are characterized by their curiosity and eagerness to find the answers to their questions and gain knowledge about something. Moreover, in the modern era, the number of people that turn to the Internet for about anything is vastly increasing. With that in mind, we conclude that Question Answering Systems (QASs) are of great importance to our lives. A Question Answering System is an information retrieval system that aims to automatically provide an answer to a (natural language) question submitted by a human [64]. Since 1961 and the first QA system, Green et al.'s BASEBALL [30], QASs have developed a lot, always improving and trying to fit the needs of the people.

Moreover, it is true that QA models can be machine or deep learning models as well as have a more rule based nature. QASs aim to understand the structure of a language and its intricacies. Along the rich history of Question Answering Systems various Machine Learning models have been used for question classification purposes like: Support Vector Machine (SVM) classifiers, Bayesian classifiers, Maximum Entropy models, etc. Moreover, there is a plethora of Deep Learning models that have been deployed for NLP tasks, some of them include: the mostly used Recurrent Neural Networks (RNNs) like LSTMs and GRUs as well as the various Transformers models using attention mechanisms that managed to improve QA tasks a great deal. Also, in recent years, the

BERT-BiLSTM-CRF-based NER method has come into focus in research. Lastly, we have to highlight the fact that lately it has been noticed that Graph neural networks (GNNs) have shown great results when working with data that are represented in the form of graphs.

As mentioned earlier, with the constant increase of digitized information and the need of so many people for quality answers and retrieved information, QASs have been combined with an emerging technology of great potential called Knowledge Graphs. Together they make a Question Answering System based on a Knowledge Graph (KGQA).

In this thesis we have conducted an extensive research on both of the technologies already mentioned; we have researched Knowledge Graphs (KGs) as well as Question Answering Systems (QASs) in order to be able to create our own Question Answering System that will be based on a (domain specific) Knowledge Graph. The main aim of this work is to explore the possibilities and potential of Knowledge Graphs combined with a QAS for a specific domain of interest. As a domain we chose Tourism because of its significance in both the world's economy – contributing 10.3% to global GDP [66] – and the quality of people's lives. As for our KG use case we decided to focus on creating an in depth Tourism Knowledge Graph about the Greek island of Santorini.

In the next 2 sections we will be sharing the research problem that motivated us (Section 1.1) as well as the detailed structure of our thesis (Section 1.2) .

1.1 Research Problem

In this Section we will discuss the research problem that motivated us through this research study. A lot of people nowadays, rather than being more spontaneous, they prefer to do a lot of research and planning before going on a trip. Making sure that you have a great tourist experience by doing all the things that interest you and visiting the places of interest that you like as well as not falling into tourist traps plays a very significant role for the tourist.

We have to highlight the fact that, nowadays, in order to find the tourism information that we need, we have shifted from using Traditional Travel Guides to googling and browsing through various sites. Furthermore, tourists have a wide variety of sources for finding information about their next travel destination such as Tourism Board Websites, Tripadvisor, Forums, Travel Blogs and Travel Magazines. It is true that there is a lot of tourism information scattered on the web to the point where there is an overabundance of it. Additionally, it is quite burdensome and confusing to browse through different sources of related or unrelated information in order to find what you need.

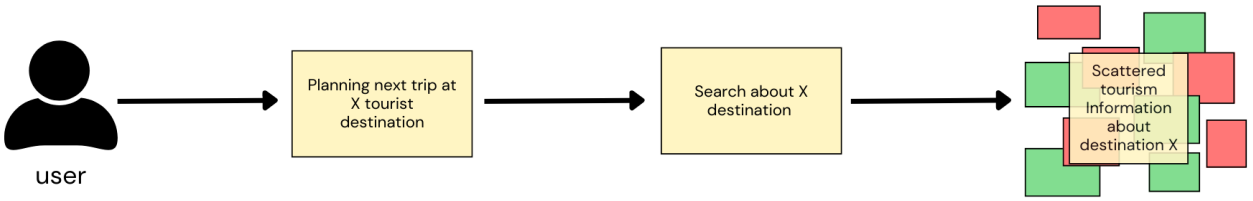

Figure 1. Illustration showing the usual process a user must go through when planning their next trip

The need to have a great tourist experience by being well informed about your next travel destination without the confusing and time consuming process of it, is the core of our research problem. Creating a Question Answering system based on a Tourism Knowledge Graph offers the user the ability to question a system with a centralized and semantically rich knowledge about their next travel destination and as a result help them better plan their itinerary.To be more precise, we aim to create a system that will understand the user's natural language question about his next tourism destination and it will return an answer that would be knowledgeable and orderly as well as be rich in information. In order to achieve that and create a more precise and efficient system overall, we have used the appropriate approaches and technologies. We have designed our own ontology that has great semantic significance and offers to organize our KG in a useful way, we, also, have deployed a knowledge graph database in order to record and discover the intricate relationships between the pieces of knowledge, and lastly, we have researched and implemented different machine learning and deep learning algorithms in order to better understand and analyze the natural language.

1.2 Thesis Structure

In detail, the present thesis is structured as follows:

This chapter aims to introduce us to this study topic by briefly defining its key features and by presenting the research questions as well as the research problem that motivated us. Furthermore, this chapter includes a detailed presentation of the structure of this work.

The **Second Chapter** is a bibliographic review that aims to provide us with all the background information we are going to need to know in order to better grasp the rest of the thesis. The first section of the second chapter focuses on Knowledge Graphs and does a thorough literature review of KGs; starting with the long history of Knowledge Graphs and moving on to answer any questions about RDF Terminology, KG creation, hosting, different types of Knowledge Graphs, the significance of this technology and, lastly, the variety of applications and industries that KGs are present. In the meanwhile, section 2 of chapter 2, intends to present a brief bibliographic review of Question Answering Systems. This section provides us with useful information about QAS by mainly referring to their historical context and the plethora of QAS categories that currently exist.

Moving forward to **Chapter 3**, we will be able to have a close look at a Knowledge Graph's specific domain, Tourism. We are going to discuss a variety of related work that has been researched regarding Tourism domain Knowledge Graphs. We divide this section into two subsections, one focusing on related work done on Tourism Domain KGs and the other focusing on a more specific application of Tourism Domain Knowledge Graphs where they are used for Question Answering.

In **Chapter 4**, we focus on the Tourism Knowledge Graph that we want to create with the use case of Santorini. The two main points we are going to be discussing in Chapter 4 is firstly, the ontology we created for the purposes of building our Tourism Knowledge Graph and secondly, the technical part of creating and populating our KG. In addition, we go on to record the whole research process we had to go through regarding the creation of our TKG.

On the other hand, **Chapter 5**, focuses on the second research topic of this thesis, Question Answering Systems. In Chapter 5, consisting of 5 Sections, we are trying to analyze the process of building our QAS. We manage to do that by breaking down the system into 4 important parts that we briefly present in the first section along with the whole Question Answering System's overview. Afterwards we break down each part in each section.

We begin by presenting the Template Library we created for the purposes of building our template-based, closed domain QAS. In Section 3 and 4 we analyze the Question Classification and Template Matching subsystems respectively, that implement state-of-the-art technologies like BERT embeddings and SentenceTransformers in order to help bring the QAS into life. Last but not least, we describe the last phase of our Question Answering system, the Answer Retrieval, focusing in retrieving and "shaping" the cypher query accordingly before using it to query the Tourism Knowledge Graph, as well as presenting the user with the retrieved answer to their question plus a recommended follow up question they might be interested in based on their original question.

In **Chapter 6**, we focus on presenting the experiments we conducted on the QAS we have built as well as their results. We divide the experiments into 3 sections; namely in the first section we have the experiments on the Question Classification subsystem where we have experimented with different preprocessing methods and parameters whereas in the second section we focus on the experiments we have conducted on the Sentence Similarity part of the Template Matching subsystem where we have experimented with different preprocessing methods as well as different SentenceTransformers models. Section 3 presents the experiment we have conducted on the Template Matching subsystem where we present the accuracy scores regarding each of its subsystems for the best and worst version.

Lastly, Chapter 7, concludes this thesis by summarizing the processes and findings of this research work. We close this study by mentioning any future work that could be done in order to improve our Question Answering system based on a Tourism Knowledge Graph.

Chapter 2

Background

In this chapter, we will analyze all the **background information** that we are going to need in order to properly understand the "tools" that we will be using in the next few chapters. This chapter should be able to inform and familiarize the reader with both of the technologies being discussed. The aim is to answer all the questions surrounding the terms **"Knowledge Graph"** as well as **"Question Answering System"**; namely these questions may vary from *"What is a knowledge graph and how it came to be"* to *"When was the first Question Answering System created and what types of Question Answering Systems are out there"*.

In particular, **Section 2.1** aims to introduce us to the **Knowledge Graph** – what it is, what is the history behind it, why are knowledge graphs considered an important technology and how can it be used – are some of the information being presented through this section and its subsections. At the same time, **Section 2.2** aims to bring light to the technology of **Question Answering Systems** and review their long history as well as the ways they differentiate from one another.

2.1 Knowledge Graphs

Knowledge Graphs (KGs) have come into focus in both research and business as an emerging technology, and have often been associated with the Semantic Web technologies, linked data, large-scale data analytics and cloud computing [21]. While trying to properly define the term *"Knowledge Graph"* we have come upon the realization that the term has been given many various definitions as a result of the considerably big publishing interest around the area in recent years. Below we are attempting to list some of the definitions that were given towards Knowledge Graphs over the years as well as give our own definition inspired by them.

According to ***H. Paulheim,*** a **knowledge graph can be defined as** *a description of real-world entities and their interrelations, organized in a graph, able to define possible classes and relations of entities in a schema, allows for potentially interrelating arbitrary entities with each other and covers various topical domains* [44].

Furthermore, **another definition** of the term "Knowledge Graph" has been given by ***Farber et al*** stating that: *"We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple (s, p, o) is an ordered set of the following RDF terms: a subject $s \in U \cup B$, a predicate $p \in U$, and an object $U \cup B \cup L$. An RDF term is either a URI $u \in U$, a blank node $b \in B$, or a literal $l \in L$."* [22].

We will go into more detail about the subject of RDF and RDF Graphs again in the following subsections but for now, we will attempt to give **a simple and inclusive definition of Knowledge Graphs (KGs)**.

We refer to a Knowledge Graph as a graph of interconnected data which consists of three key elements:

- The **first key element** is the **real-world entities of interest** - for example, people, movies, events, situations, etc.- that are represented as **nodes**.
- The **second** is the **relations** that are the **edges** connecting two nodes, illustrating the **relationship** between them.
- And the **third key element** would be the **attributes** which are the **properties of the entities** acting as the characteristics that describe them.

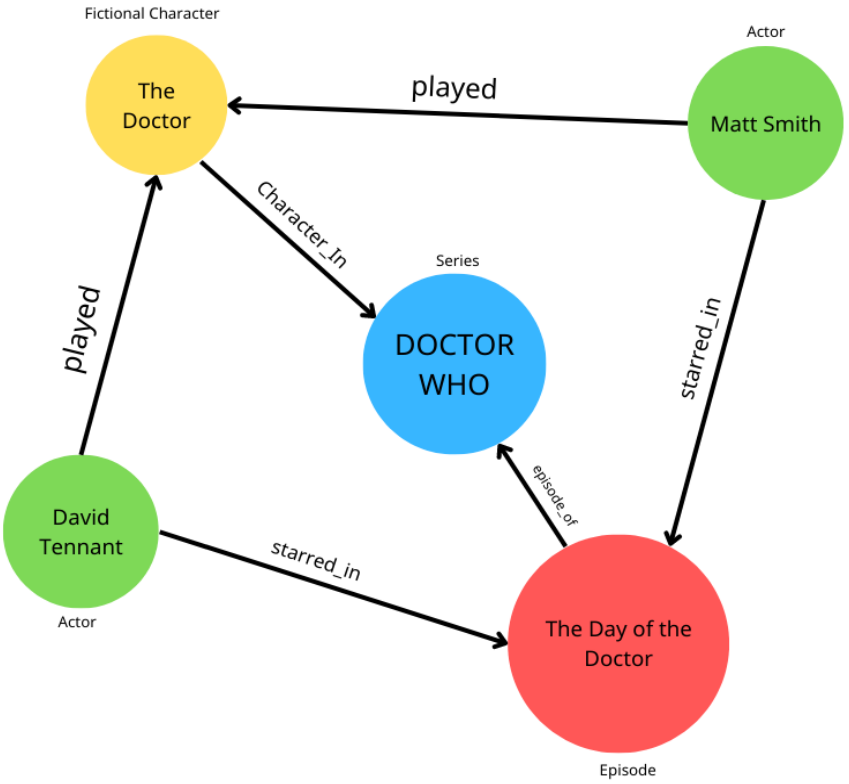

Figure 2. Sample knowledge graph. Nodes represent entities, edges represent relationships and property labels represent attributes

After providing the definition we will need to analyze some of the **key ideas behind Knowledge Graphs.** It is true that the power of the Knowledge Graph derives from the nature of its entities, previously referred to as **real-world entities of interest.** A Knowledge Graph is fundamentally a collection of several layers of knowledge related to an entity [3].

Furthermore, a Knowledge Graph is often referred to as a Semantic Network. One thing that is important about KGs is their ability to integrate various heterogeneous information and knowledge aggregated from a number of sources in order to create a "rich" insightful and diverse network

capable of decision-making, deriving new information as well as other applications for which we will be discussing in ***Subsection 2.1.5.***

2.1.1 Historical Context

In order to trace the **origins of Knowledge Graphs** we will have to travel back to **1735** when Swiss mathematician **Leonhard Euler invented graph theory** in the process of solving the Königsberg bridge problem. Regarding **Knowledge Graphs,** over the years the term was generally used, often unrelated to its modern meaning. Even though **Knowledge Graphs**' popularity has been significantly raised in the last few years due to the introduction of **Google's Knowledge Graph** in 2012 (*see more in* ***later subsections***), its foundation lies before that, dating back to **1973** with *Schneider's* paper on computerized instructional systems for education [57].

The **first knowledge graph** was created in **1985** and it was called **Wordnet.** It can be described as a **Natural Language Processing (NLP)** database of semantic relationships that share structural similarities between the words, despite the language that they are expressed [32]. After that, in **2005**, Marc Wirk founded **Geonames**, a knowledge graph that represents relationships between different geographic names and locations and associated entities. Two years later, in **2007**, **DBpedia and Freebase** were founded as graph-based knowledge repositories for general-purpose knowledge. Even though neither of them was described as a "knowledge graph", they evolved to be one. We will talk more about them **in subsection 2.1.3**. As mentioned earlier since **Google** introduced its **Knowledge Graph** in **2012**, the term got huge amounts of attention. It is important to notice that Google's Knowledge Graph is built based on the KGs that we mentioned above; DBpedia and Freebase.

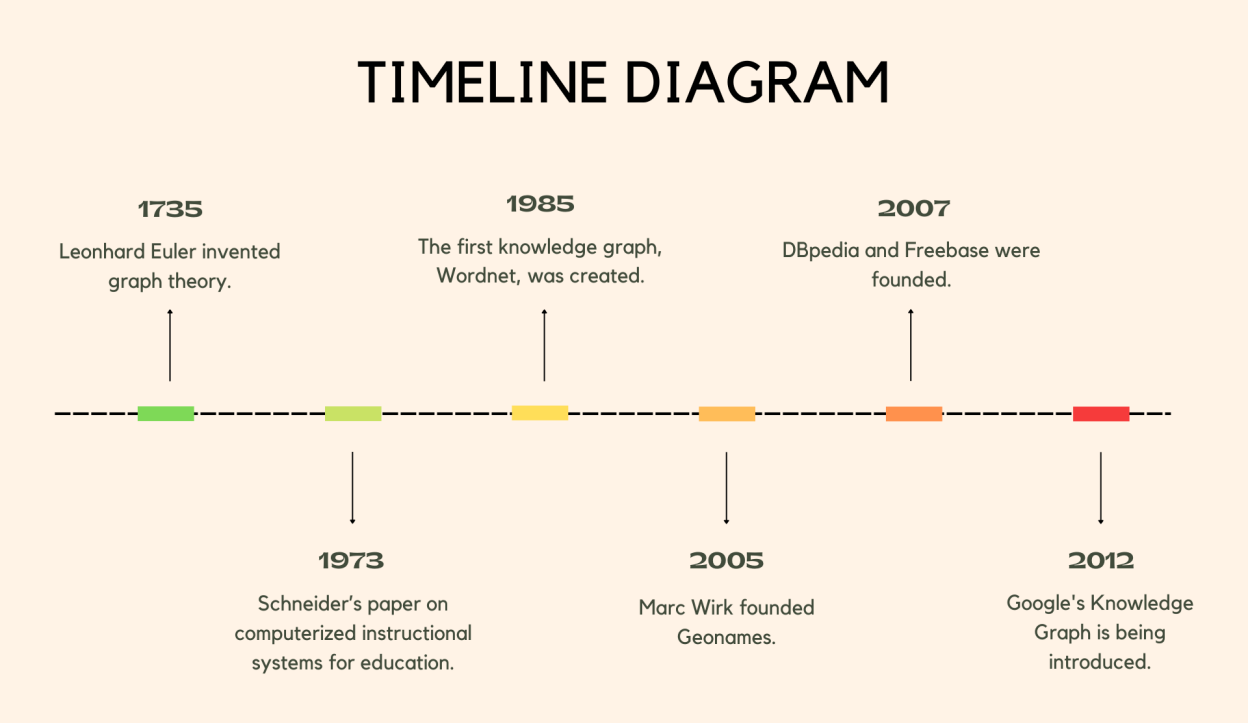

Figure 3. Timeline diagram of Knowledge Graphs' brief history

2.1.2 RDF Terminology and KG Creation & Hosting

In this subsection, we are going to be taking the time to properly explain all the **RDF Terminology** that we are going to be using in later subsections. Additionally, we are going to be discussing the **various KG creation techniques** that are being implemented nowadays. Lastly, we will be referring to **a number of different hosting services** where someone can **host and query** their Knowledge Graph.

Undoubtedly, **Knowledge Graphs** are indissolubly linked to the **W3C (World Wide Web Consortium) RDF** which stands for **Resource Description Framework**. Through the last few paragraphs, we have come across references of the term RDF but until now we didn't really go into detail about it; what it is and how it can be used. I will start addressing the above questions by **properly defining** the term "**RDF**":

According to the *RDF 1.1 Primer*: *"The **Resource Description Framework (RDF)** is a **framework** for expressing information about **resources.**"* Resources can be anything from documents, people, physical objects, etc. Due to its nature, **RDF** can be used to **describe data that is highly interconnected**. **RDF statements** have a **three-part structure** (subject-predicate-object) consisting of resources that have URIs appointed to them. Describing data in **RDF triples** allows the information to be more **easily represented** and **interlinked** while managing to maintain all its **expressivity**.

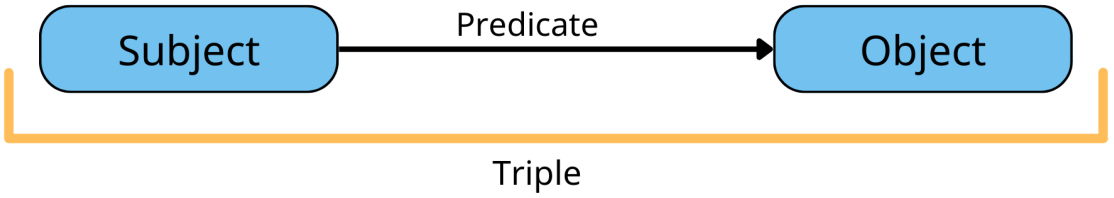

Figure 4. Simple representation of a RDF Triple (s, p, o)

In *Figure 4.* we can see what an RDF triple looks like, containing two nodes – the source node called **Subject** and the target node called **Object –** and a triple connecting them (**Predicate**). It's important to notice the fact that the same node can be found in the subject position in one triple and in the object position in another triple.

Moreover, regarding the **RDF data model** we have to mention that **RDF nodes** come in **3 different types**:

1. **Internationalized Resource Identifiers (IRIs)**: IRIs, as we mentioned earlier, **identify resources.** The concept of IRI is a generalization of URI (Uniform Resource Identifier), which in turn is a generalization of the well-known URLs (Uniform Resource Locators). As a result, an **IRI can come in the form of a URI or a URL**. IRIs can appear in the position of the object, subject, or predicate. A collection of IRIs is called an **RDF Vocabulary.** Some of the most **common RDF Vocabularies** used worldwide are RDF and RDFS Vocabularies, OWL, Dublin Core, Friend of a Friend (FOAF), SKOS and schema.org [14].

2. **Literals:** Literals are values that are not IRIs and are used to represent **datatypes like strings, numbers and dates.** It is important to notice that literals can only be in the **object** position in an RDF triple.

3. **Blank Nodes:** Additionally, there are Blank Nodes (also known as "bnodes") that allow us to refer to a resource without having to use a global identifier.

Below we will be giving an **example of RDF triples** (using pseudocode) in order to better understand how it can be illustrated. Each element is enclosed within an angle bracket (<>).

```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
```

Example 1: Sample triples [49]

Nevertheless, in need of an appropriate way to **load graphs into a system**, there came to be a number of different **syntaxes in order to serialize an RDF graph**. Below we are going to be referring to some of those syntaxes:

- Firstly, there is the **Turtle family of RDF languages** including:
  - **N-Triples**
  - **Turtle**
  - **TriG**
  - **N-Quads**
- Another syntax is **JSON-LD.**
- Also we can't disregard **RDFa.**
- Lastly there is the **RDF/XML** syntax.

Regarding the **Turtle RDF** format, besides being based on the N-Triples syntax and following its format, it also allows the use of **namespace prefixes, lists** and **shorthands for datatyped literals** [49]. A **prefix** is a way to **shorten a long namespace** (IRI) into a shorter local name. In addition to this, **Turtle RDF** offers the ability to have **multiple predicates**, **multiple objects** and **types**. We have to notice the fact that the letter **a** can be used instead of the **rdf:type**.

Furthermore, as for the **creation of a Knowledge Graph**, **Hogan, A. *et al.*** mention two approaches by which someone can create a Knowledge Graph; namely, the techniques are as follows:

- **Human Collaboration:** In this approach, we can notice that the **creation process involves humans** and the "human" touch. The contributions assisting in the creation of the graph can derive from various sources, such as in-house editors, crowd-sourcing platforms, feedback

mechanisms and others. However, every approach can have its downside. That is the case here, too, with the "human" aspect of the approach often acting in a negative way [32].

- **Text Sources:** Alternatively, there is the more **automatic Machine Learning fueled approach** that utilizes techniques such as **Natural Language Processing (NLP)** as well as **Information Extraction (IE)**. The core tasks that take place in text extraction are Pre-processing, Named Entity Recognition (NER), Entity Linking (EL), Relation Extraction (RE). Of course, there can be several different task sequences that a framework can follow [32].

Lastly, regarding **Knowledge Graph Hosting** there is a plethora of databases where someone can store graphical information. Depending on the **type of the Graph** ***(see more in Subsection 2.1.3)***, there are different graph databases as well as **query languages**. For **RDF Triple Stores** some of the **most popular databases** are Oracle Spatial and Graph with Oracle Database 12c, AnzoGraph DB by Cambridge Semantics, AllegroGraph, Stardog, OpenLink Virtuoso, GraphDB™ by Ontotext [37]. RDF Triple Stores use **SPARQL** as their standard querying language in order to extract information from the database. On the other hand, there are **Property Graphs** whose most adopted query language is **Cypher** and some of the most commonly used databases are Neo4j and OrientDB.

2.1.3 Types of Knowledge Graphs

As mentioned earlier, in recent years knowledge graphs have managed to intrigue both companies and researchers. As a result, today many **Open Knowledge Graphs** are being published. At the same time, many **Enterprise Knowledge Graphs** are being created as well.

While researching **Open Knowledge Graphs** – *by open here we refer to the Open Data Philosophy that anyone can freely access the data* [32] – we come across a plethora of **Cross-Domain KGs** such as DBpedia, YAGO, Freebase, Wikidata, and a variety of **Domain-Specific KGs**; namely the most prominent domains as listed by **Hogan, A. *et al***. are *media, government, publications, geographic, life sciences, user-generated content, cultural heritage, music, law, theology and even tourism [*32].

Regarding the **Enterprise Knowledge Graphs,** one thing that characterizes them is their quite **large size**. In recent years Enterprise KGs are indeed being deployed by many **companies** (Facebook, Microsoft, Amazon, etc.) in many **industries**, like Web Search, Commerce, Social Networks, etc. In ***Section 2.1.5*** we will study more about the applications these types of KGs may have.

So if we were to group KGs into groups we could separate them **based on the specificity of the domain** thus having **Domain-Specific Knowledge Graphs** that are focused on a particular domain of interest or **Cross-Domain Knowledge Graphs** in which case the knowledge derives from multiple domains and the KG is able to represent a broad diversity of entities and relationships [32].

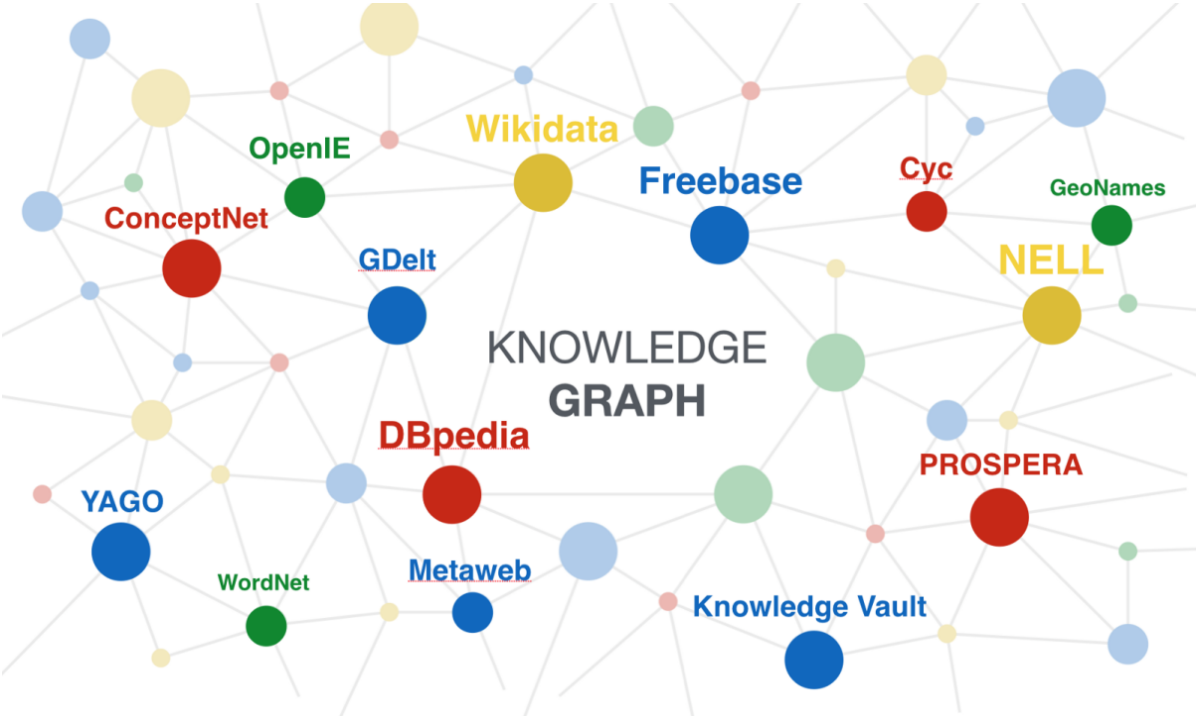

Figure 5. Some of the most popular Cross-Domain Knowledge Graphs [38]

In *Table 1* we have analyzed the sizes of some of the most popular Knowledge Graphs in order to understand how much larger they are compared to each other. **DBpedia** is a project which aims to extract structured content from the semi-structured data available in the Wikipedia project [17]. **YAGO** (Yet Another Great Ontology) is a knowledge base that extracts information from both Wikipedia and other sources like WordNet and GeoNames [68].

Freebase was a broad collection of data harvested from a variety of sources and what made it different from the above projects was its collaborative nature; all the data was mainly composed of its community members' contributions [25]. **Wikidata** is also a collaborative, multilingual knowledge graph whose main purpose is to provide structured data to Wikipedia as well as other wikis in the Wikimedia movement [62]. **Google's Knowledge Graph** – which we know very little about – is a knowledge base that Google uses to power its **search results sidebar panel**.

Name	Instances	Facts	Types	Relations
DBpedia (English)	4,806,150	176,043,129	735	2,813
YAGO	4,595,906	25,946,870	488,469	77
Freebase	49,947,845	3,041,722,635	26,507	37,781
Wikidata	15,602,060	65,993,797	23,157	1,673
NELL	2,006,896	432,845	285	425
OpenCyc	118,499	2,413,894	45,153	18,526

Name	Instances	Facts	Types	Relations
Google's Knowledge Graph	570,000,000	18,000,000,000	1,500	35,000
Google's Knowledge Vault	45,000,000	271,000,000	1,100	4,469
Yahoo! Knowledge Graph	3,443,743	1,391,054,990	250	800

Table 1. Numerical Overview of some Knowledge Graphs [44]

Nevertheless, we can always **categorize a Knowledge Graph by its data model**; that is **RDF Graph,** also referred to as RDF Triple Store, or **Property Graph,** also called Labeled-Property Graph. As we saw earlier by the term **"RDF Graph"** we refer to a graph that consists of a set of **RDF triples**; ***subject, predicate, object*** [22]. An RDF Graph is created with the use of the RDF Syntax (***see Subsection 2.1.2***). Some of the **advantages** of implementing **RDF Graphs** are:

1. First and foremost, one of RDF's advantages is its high **Interoperability** that allows the consolidation of many different Knowledge Graphs thus facilitating the reusability of the KGs.
2. Secondly, we have to mention the advantage of having a **standard framework** for representing data while maintaining high expressivity.
3. Lastly, we can't overlook RDF Graphs' **extensible** and **flexible nature** through which users have the ability to introduce something new (nodes and relationships) to the graph without having to build it anew.

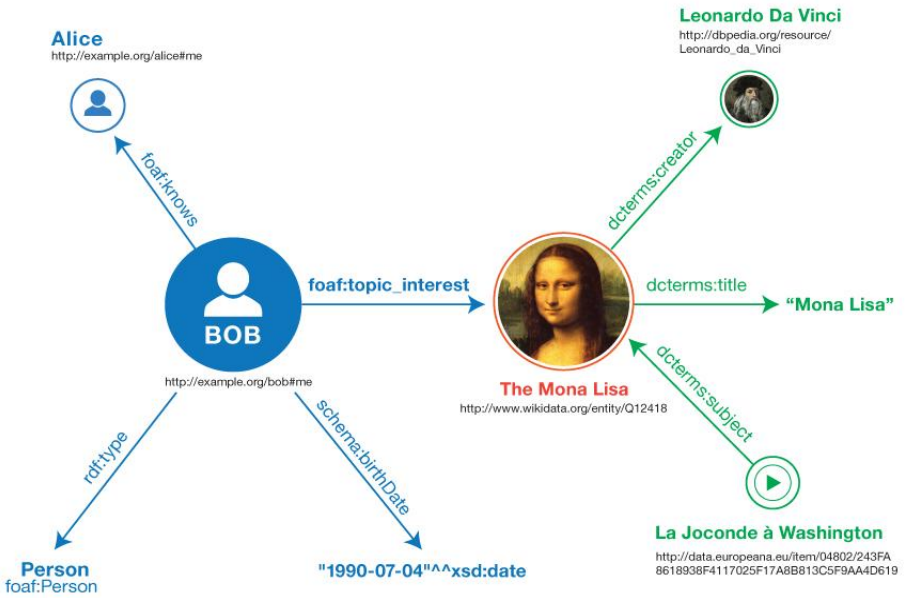

Figure 6. Example of an RDF Graph [49]

The above figure (Figure 6.) portrays an RDF graph that manages to capture the "richness" and semantics of the interconnected data. The example above is using the N-Triples serialization format in order to illustrate the **interest** *(foaf:topic_interest)* that **Bob** *(http://example.org/bob#me*) shows for **The Mona Lisa** *(http://www.wikidata.org/entity/Q12418)*. This is an RDF triple where "*foaf:topic_interest"* is the **Predicate** linking the two resources, "*http://example.org/bob#me*" (**Subject)** and "*http://www.wikidata.org/entity/Q12418*" (**Object).** There can be **three** kinds of nodes in an RDF graph: **IRIs, literals, and blank nodes** [48]. In the graph above, we can see IRIs (e.x. http://example.org/bob#me) as well as literals (e.x. "Mona Lisa").

On the other hand, **Labeled Property Graphs (LPGs)** are another kind of Knowledge Graph. LPGs take a much different approach to how they **depict knowledge**. The data here is organized as **nodes, relationships, and properties.** It is important to mention that what makes Property Graphs differ is the fact that both the **nodes** (entities), **as well as the relationships,** that connect them can be described by (zero or more) properties.

Some of the main **advantages** of using **Labeled Property Graphs** are:

1. LPGs are generally much more **simple** and **easy for new users** to create and use.
2. Moreover, another known advantage of property graphs is their proneness to **detail,** having properties on the relationships and nodes offers more information.
3. **Navigating and querying** a property graph can indeed be easy.

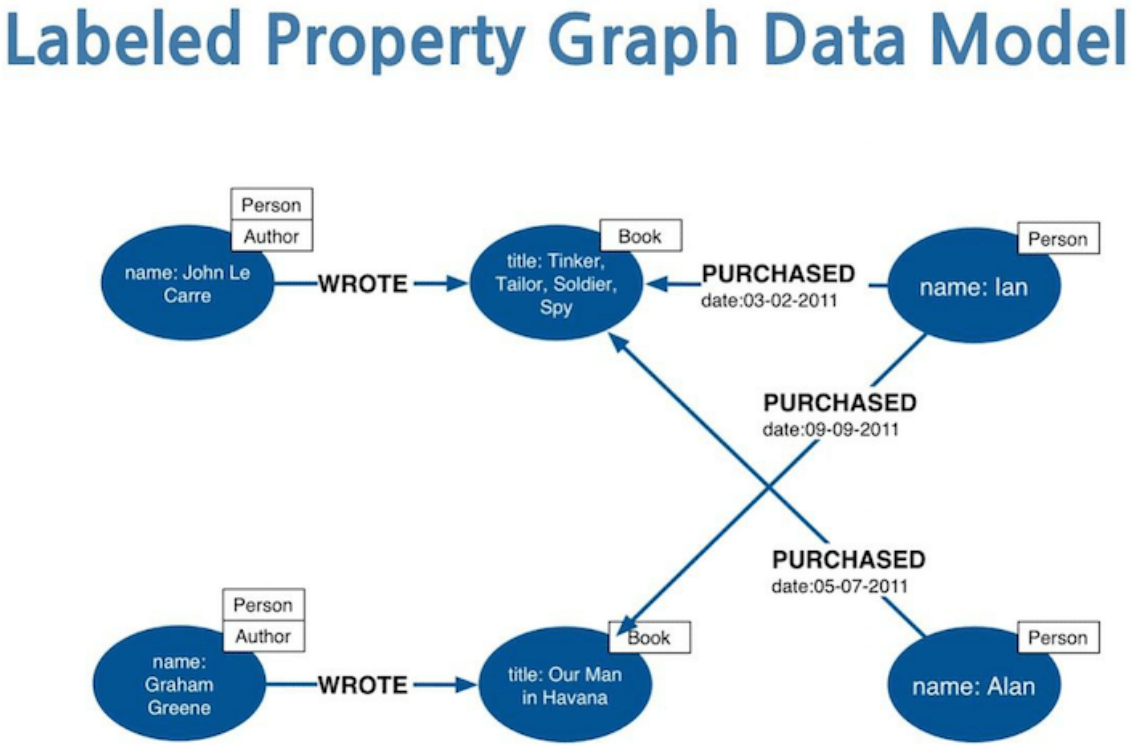

Figure 7. Example of a Labeled Property Graph [42]

In *Figure 7.* we can see how a **Labeled Property Graph** resembles. By comparing it to the graph in *Figure 6*. we can notice how they differ from one another on a fundamental level. At first sight, the Labeled Property Graph shows many similarities with the RDF graph regarding its connectivity and data richness. However, **Labeled Property Graphs** are characterized by their "labels" and their "properties". On the graph above we can see that every node has its own **label** (e.x. Person, Book, etc.) while some of the edges carry **properties** (showing the date of the purchase).

2.1.4 The importance of Knowledge Graphs

The **interest** as well as the **research** that is being conducted around this area in recent years can't go unnoticed. Even though **Knowledge Graphs** can often be described as a **research area** in its **infancy**, we can place KGs among the promising emerging technologies in the science of data. Therefore, we can assume that Knowledge Graphs will play a **significant** role in **Artificial Intelligence** and Data Science in the years to come.

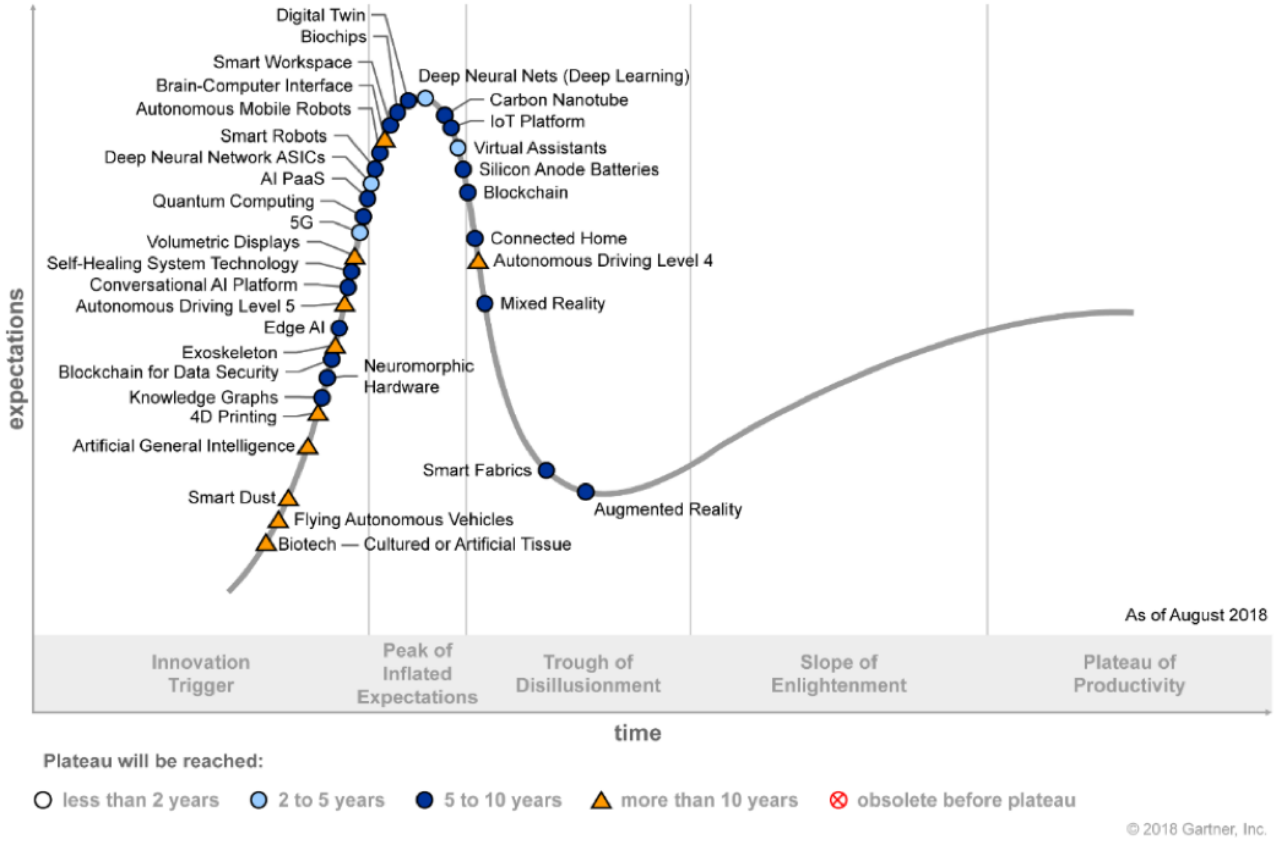

Figure 8. Hype Cycle for Emerging Technologies, 2018 [26].

The implementation of **Knowledge Graphs** by businesses and organizations can play a **significant** role in their **growth**. Every day a plethora of companies are moving towards using Knowledge Graphs, as it can be a very **valuable tool** for solving many business problems, such as **decision making** and **harnessing data insights**. If we take a look at our **daily lives** we will come across **several examples** of knowledge graph technology being exploited; some of them are Alexa, Google Assistant and Siri (for more applications see subsection 2.1.5).

The true potential of the Knowledge Graphs can be found in the way their integrated data are **connected**. Because of that, **Knowledge Graphs** are often compared to a **human brain**. These complex mind maps emphasize on the representation of **intricate relationships between entities** while also focusing on **uncovering the underlying knowledge.** In order to understand why someone should use a **Knowledge Graph** instead of a traditional **Relational Database**, below we are going to be **highlighting the differences** between them.

Factor	Knowledge Graph	Relational Database
Schema	Schema-free. Flexible.	Schema-driven. Data Structure is pre-defined.
Performance	Faster than relational databases.	Relatively slower than Knowledge Graphs. Require lots of joins.
Maintenance	A lot easy, as they are schema-free	Difficult and often cumbersome, as minor changes could affect the entire structure

Table 2. Overview of the differences between Knowledge Graphs and Relational Databases

First and foremost, compared to KGs, Relational Databases are not able to either represent nor harness the **complex relations between data** in the same way that knowledge graphs do. This is due to the fact that **knowledge graphs** have **no fixed schema** which automatically makes them more flexible in terms of integrating data with different structures. On the contrary, **relational databases** are **schema-driven,** meaning that all data entries must follow a specific, pre-defined structure.

Secondly, the **performance** of knowledge graphs when **running queries** is indeed much **faster** compared to relational databases because of the way the data are structured and **interlinked together**. **Querying a relational database** requires a lot of **joins** resulting in slower performance.

Finally, the **last difference** between relational databases and knowledge graphs lies in their **maintenance**. KGs are **much easier to maintain** because, as we have already noticed, they have no specific schema and compared to relational databases, here we don't have to worry about the "side effects" of storing additional data.

2.1.5 Applications

In this subsection we are going to be listing several different **applications** using **Knowledge Graphs** as well as mentioning the **plethora of industries and companies** that in recent years have moved towards using KGs. Recent **applications** using a KG include: **Decision-making**, **Fraud Detection** as well as **Anomaly Detection**, **Recommendation Systems** [12], **Question Answering** (see more in Section 3.1.2), **Information Search & Retrieval** [35], KG powered **Drug Discovery** [43], implementing **Personal Agents** [15], enhancing **Targeted Advertising** [9], increasing **Transport Automation** [31].

As we have already seen, a Knowledge Graph can be a really **valuable tool** for businesses across **many industries** such as: **Web Search**, **Commerce**, **Social Networks**, **Finance, Healthcare**, **Transport**, **Oil & Gas**, **Entertainment** [32]..

As we mentioned in Subsections 2.1.1 and 2.1.3, **Google's Knowledge Graph** being introduced in 2012 highlighted the **potential** of this technology. With the phrase ***"things, not strings",*** Google managed to reinvigorate the general interest over Knowledge Graphs. Since then, many major companies have started to implement this technology; some of them being: **Google**, **Microsoft** [8], **Yahoo**, **Apple**, **Amazon** [41], **Meta**, **EBay** [15], **LinkedIn** [9], **AirBnB** [12], **Uber** [24]. amongst others.

2.2 Question Answering Systems

In Section 2.2 we aim to give some background context about Question Answering Systems in order to gain a better understanding of them; What is **Question answering?** How do QA systems differentiate from each other? What is the origin and history of **Question Answering Systems?** What are some useful QA applications? Through the next few subsections we are going to try and answer all of the above questions.

First and foremost, let's start by giving a definition of what a Question Answering System is: **Question answering (QA)** is a subfield of **Natural Language Processing (NLP)** and **Information Retrieval** (IR). The main task of **Question Answering Systems** (QASs) is to automatically **provide an answer to a** (natural language) **question** submitted by a **human** [64].

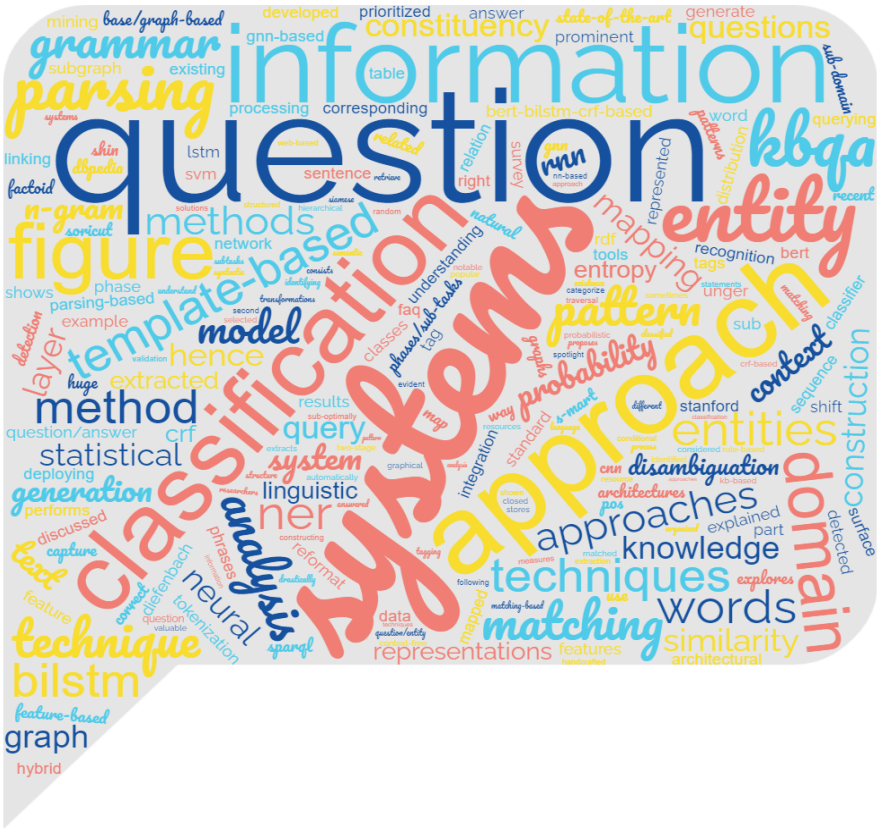

Figure 9. Question Answering Abstract Word Cloud

It is true that curiosity is a fundamental human trait that characterizes us and as such we are always full of questions that seek to be answered. Furthermore, nowadays a vast number of people turn to the Internet for answers as it is the main source of information today. Therefore, we conclude that Question Answering Systems are of great importance to our lives.

Their importance can be seen through the plethora of useful applications a QA system has. Some of the most typical applications of QASs are related to **Customer Support** and **QA bots**. The use of smart virtual assistants that aim to assist customers creates a valuable tool for businesses. Additionally, chatbots for question answering as well as FAQ bots are another famous example of question answering systems. We, also, have to notice the case of **Search Engines** using QAS, such as Google. Google deploys a QA system in order to come up with questions and answers. More specifically, when someone clicks on a question, the list increases because new questions appear that are a lot more similar to the one that they clicked on [6]. Lastly, a few other applications of Question Answering Systems include: Data Analytics, Market Research and Fact Checking.

In the next two subsections we are going to present the Historical Context behind QASs (Subsection 2.2.1) as well as analyze the different types of Question Answering systems (Subsection 2.2.2).

2.2.1 Historical Context

This subsection aims to briefly present the rich history of question answering systems by providing us with historical information going back to the early systems. The development of QA systems began in 1961 with Green et al.'s **BASEBALL** [30]. This QA system was a simple closed domain information retrieval system that focused on answering questions regarding American League baseball games.

Another early QA system we need to mention is **LUNAR** [65]. LUNAR was a QA system that helped geologists on the Apollo moon mission [10]. We must highlight the fact that both of the previously mentioned systems resemble the first chatbot programs – ELIZA and DOCTOR – as they share similar techniques [64]. Furthermore, around the late 1960s and early 1970s **SHRDLU** was developed by Terry Winograd. This was a very successful QAS that simulated the operation of a robot in a toy world while also providing the chance to question the robot about the state of the toy world [64].

Finally, reaching today, we must notice that Question Answering Systems have seen significant progress and change. Specifically, IBM Research has created **Watson** [23], a deep learning QA system that managed to surpass human-level intelligence when it competed against the best contestants on "Jeopardy" and won [10].

2.2.2 Types of Question Answering Systems

While conducting this background research on Question Answering Systems we came to the conclusion that there are multiple different ways that someone can categorize a QAS. Below we will try to do a detailed presentation of these categories.

Firstly, one way to differentiate QA systems is by their domain. QASs are divided into open domain or closed-domain [10]. Open domain QA Systems answer questions regarding nearly anything in the general world knowledge [64]. On the other hand, according to Allam et al. *"Closed domain question answering deals with questions under a specific domain (music, weather, forecasting, etc.)"* [5].

Secondly, according to Zope et al. we can generally divide QA system approaches into four important classes [72]:

- ➔ **Linguistic Approach**
- ➔ **Statistical Approach**
- ➔ **Pattern Matching Approach**
- ➔ **Hybrid Approach**

The Linguistic Approach implements a variety of different techniques such as tokenization, POS tagging, and parsing. Additionally, the Pattern Matching Approach includes Surface Pattern based Techniques and Template based Techniques [72].

Lastly, we must highlight the fact that a Question Answering System can be classified based on its architecture. When developing a question answering system, there are three main architectures; **Text-based, Knowledge-based (KBQA)** and **Hybrid**. The Text-based architecture relies on unstructured data such as text documents in order to find the most relevant answer of all possible answers. On the other hand, knowledge-based architecture relies on **structured data** stored in a knowledge base [10]. We must mention an interesting and upcoming subset of KBQA systems that is Knowledge Graph Question Answering Systems (KGQA).

In addition, all the QA knowledge-based systems can be classified in four different architectures; **Semantic Parsing-based, Subgraph Matching-based, Template-Based** and **Information Extraction (IE)-Based.** We must highlight this new and promising architecture named Graph neural network (GNN) that according to Zope et al. *"manages to show better results when processing the data represented in the form of graphs"* [72].

Regarding the KBQA systems, Diefenbach et al. identified five tasks in the QA process [19]. These tasks are:

- **Question Analysis**
- **Phrase Mapping**
- **Disambiguation**
- **Query Construction**
- **Querying Distributed Knowledge**

Entity Linking is also a really important task in a QA system. It consists of Named Entity Recognition (NER) and Disambiguation. Many great technologies have been deployed in this task with some of them being Conditional Random Field (CRF), various RNN models, BiLSTM etc. as well as the BERT-BiLSTM-CRF-based NER method which has state-of-the-art results.

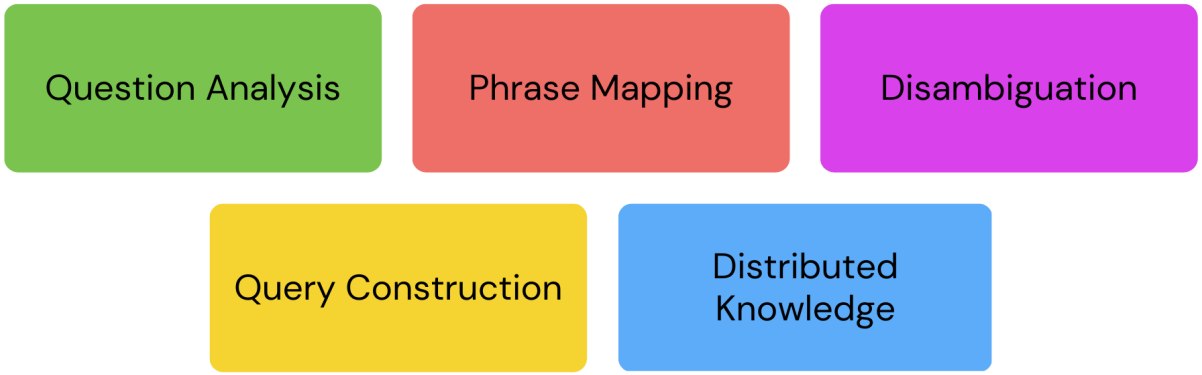

Figure 10. The five tasks in the Question Answering process

Chapter 3

Knowledge Graphs and Tourism

Through this chapter, we aim to describe how the Tourism domain can be a promising Knowledge Graph domain. As we have already mentioned in Subsection 2.1.3. Knowledge Graphs can be separated into **Domain-Specific Knowledge Graphs (DKGs)** and **Cross-Domain Knowledge Graphs.** DKGs focus on a specific field, for example Tourism-domain Knowledge Graphs (TKG) focus on the Tourism field. As we have listed above there are a plethora of other domains besides Tourism that have gained a lot of research attention in recent years as well, although the potential of the Tourism Domain remains undiscovered.

Undoubtedly, Tourism plays a really important role not only in people's lives but also in the world's economy. According to the World Travel and Tourism Council (WTTC), in 2019 the **Travel & Tourism sector** contributed **10.3% to global GDP** [66], thus we realize that Tourism is an important sector with **high demand**. Nowadays, the amount of travel information available to us on the internet has become so **vast**, to the point where it confuses the traveler instead of helping them. Tourism Knowledge Graphs work towards **improving the traveling experience** of a tourist while contributing to the various stages of their trip, from the **general planning** to **finding recommended tourist attractions**.

3.1 Related Work

In this Section, we will be discussing the related work that has been conducted in recent years around **Tourism** Domain Knowledge Graphs (TKG). In order to better understand TKGs — we decided to split this section into two subsections. The first subsection focuses on Tourism KGs and the various ways they can be used. **Subsection 3.1.1** covers a wide range of papers describing the many different ways that a Tourism Knowledge Graph can be created and implemented. Whereas, the second subsection (**Subsection 3.1.2)** aims to describe the related work done around **Tourism Domain Knowledge Graphs** specifically used for **Question Answering Systems.**

3.1.1 Related work on Tourism Domain KGs

As mentioned earlier, in recent years there has been a noticeable amount of important research being conducted on Knowledge Graphs. Amongst the many different (domain-specific) KGs we will be focusing on the Tourism Domain in which great work has been conducted as well.

A large amount of research papers focus on the automated or semi-automated **creation** of Tourism Domain Knowledge Graphs with a plethora of case studies. First and foremost, **Xiao, D. *et al.*** face the challenge of the overflowing amount of semi-structured and unstructured tourist data through building a heterogeneous Knowledge Graph for Hainan Tourism. They present a systematic framework to build a TKG consisting of two **pipelines** of semi-structured and unstructured **data processing** using tourism InfoBox and deep learning algorithms respectively [67]. Additionally, **Zhang, W. *et al.*** attempt to organize the vast useful information of Chinese Tourism using a Knowledge Graph. They try to develop the Chinese domain-tourism knowledge application platform based on a three part development architecture (acquisition of tourism knowledge, knowledge fusion and knowledge completion). For the knowledge fusion part they achieve entity alignment with the help of the NLP Skip-Gram Model [71].

Furthermore, **Kärle, E. *et al***. show how they gather Tyrolean touristic data and store this data publicly in a knowledge graph while making use of the schema.org vocabulary. This is achieved through a wrapper software allowing automatic annotation by first reading the data source, mapping it to schema.org and then storing the resulting file [36]. Finally, regarding the KG building part we have to mention the paper by **Calleja, P. *et al.*** which presents DBtravel, a Spanish Tourism Knowledge Graph generated from the collaborative site Wikitravel. Their aim was to create a Tourism-Oriented KG by exploiting the structured information stored in Wikitravel entries. Through a natural language pipeline developed with the GATE framework they manage to extract the needed information [11].

Undoubtedly, a big amount of research being conducted regards the Tourism Knowledge Graphs meant for **Recommendation** purposes. Firstly, **Dadoun, A. *et al.*** proposed a Recommender System that would suggest a ranked list of destinations where travelers would like to visit that they have not yet visited. In order to achieve that, they **used a knowledge graph** as a data structure to store the information (past bookings of travelers) and also they **took advantage of its interlinked nature** in order to recommend the next travel destination to a traveler [16]. Secondly, **Yochum, P. *et al.*** paper focuses on building a recommendation model based on a Tourism Knowledge Graph. Their main purpose is to create a recommendation model based on a KG capable of recommending tourist attractions in Bangkok city. In order to achieve that they use a data aggregator to collect the data and then they store it in a centralized database. With the help of the Neo4j tool they generate the needed tourist attraction KG [70].

Lastly, **Lu, C., Laublet, P. and Stankovic, M.** propose a recommendation system that leverages a knowledge graph in order to recommend travel attractions. They focus on improving related drawbacks such as semantic poorness and city-agnostic user profiling strategy. We have to notice the fact that they constructed their Tourism Domain Knowledge Graph from existing KGs; namely Geonames, DBpedia and Wikidata [40].

Ref.	Paper Title	Year	Author(s)	KG Creation Method	Purpose
[16]	Predicting Your Next Trip: A Knowledge Graph-Based Multi-task Learning Approach for Travel Destination Recommendation	2021	A. Dadoun, Raphael Troncy, M. Defoin-Platel, G. A. Solano	The Airline Travel KG is based on the T-DNA database and is enriched with other Knowledge Graphs.	Recommender system that suggests a ranked list of destinations where travelers would like to go to.
[67]	A Practice of Tourism Knowledge Graph Construction based on Heterogeneous Information	2020	Dinghe Xiao, Nannan Wang, Jiangang Yu, Chunhong Zhang, Jiaqi Wu	Crawling semi-structured and unstructured Hainan Tourism data and extracting the structured knowledge using two pipelines, one for the semi-structured data (InfoBox) and one for the unstructured data (deep learning algorithms).	Information extraction (IE) so as to construct a Tourism-domain Knowledge Graph (TKG).
[71]	The Chinese Knowledge Graph on Domain-Tourism	2019	Weizhen Zhang, Han Cao, Fei Hao, Lu Yang	NLP Skip-Gram Model.	Chinese domain-tourism knowledge application platform.
[70]	Tourist Attraction Recommendation Based on Knowledge Graph	2018	Phatpicha Yochum, Liang Chang, T. Gu, Manli Zhu, Weitao Zhang	Tourism data in Bangkok is collected by the data aggregator and then stored in the centralized database. The Neo4j is used to generate the tourist attraction knowledge graph.	Building a tourist attraction recommendation model based on a knowledge graph for tourist attractions in Bangkok city.
[36]	Building an Ecosystem for the Tyrolean Tourism Knowledge Graph	2018	Elias Kärle, Umutcan Şimşek, Oleksandra Panasiuk, Dieter Fensel	Wrapper software.	Touristic IR system for Tirol in Austria.
[11]	DBtravel: A Tourism-Oriented Semantic Graph	2018	Pablo Calleja, Freddy Priyatna, Nandana Mihindukulasooriya, M. Rico	Natural Language Pipeline developed with the GATE framework.	Spanish tourism oriented knowledge service.
[40]	Travel Attractions Recommendation with Knowledge Graphs	2016	Chun Lu, Philippe Laublet, Milan Stankovic	Constructed from existing large knowledge graphs namely Geonames, DBpedia and Wikidata.	Selecting relevant travel attractions for a given user.

Ref.	Paper Title	Year	Author(s)	KG Creation Method	Purpose
[39]	Towards Knowledge-Based Tourism Chinese Question Answering System	2022	Jiahui Li, Zhiyi Luo, Hongyun Huang, Zuohua Ding	Used a Java distributed crawler framework to crawl multiple tourism websites, then they analyzed and extracted the information under different labels. After that, the crawled information was screened, integrated, and standardized.	Answering tourism questions with the help of a tourism question answering system that explores a large amount of information from travel websites.
[69]	Research on Tourism Question Answering System Based on Xi'an Tourism Knowledge Graph	2020	Lu Yang, Han Cao, Fei Hao, WeiZhen Zhang, Muhib Ahmad	Crawling entities from travel websites, cleaning the data and importing them into the neo4j graph database.	Focusing on the background and needs of (the tourism field) QA, researching the relevant technologies, as well as constructing a QA system based on a tourism knowledge graph.
[59]	Question answering system based on tourism knowledge graph	2021	Yuan Sui	-	Building a tourism knowledge graph hosted on neo4j while constructing a question answering system (QA).
[20]	Developing a Vietnamese Tourism Question Answering System Using Knowledge Graph and Deep Learning	2021	Phuc Do, Truong H. V. Phan, Brij B. Gupta	-	Developing a QA System for Vietnamese Tourism based on a knowledge graph and deep learning.

Table 3. Overview of Knowledge Graph Approaches on the domain of Tourism

3.1.2 Related work on Tourism Domain KGs and Question Answering Systems

In this subsection we will be referring to research papers that focus specifically on the application of Question Answering (QA) Systems. Besides recommendation systems – that we mentioned in the section above – another noticeable example of an application for Tourism Knowledge Graphs

(TKGs) is QA Systems. As we have mentioned in Section 2.2, in recent years important research has been conducted around QA Systems and especially on Question Answering using a Knowledge Graph. We have to notice the fact that Tourism QA Systems are required in order to face the lack of organized concentrated information regarding a tourist destination.

Firstly, Li, J. *et al.* proposes a framework for automatically constructing a TKG regarding tourist attractions in Zhejiang, China. The said Tourism KG was built by first crawling tourism websites (through a Java distributed crawler framework), analyzing as well as extracting the information under different labels, and then the information went through the process of screening, integration, and standardization. They developed a tourism question answering system that makes use of that TKG and by implementing **BERT** they managed to acquire the underlying information [39].

Additionally, Yang, L. *et al.* focuses on the tourism QA background and needs while researching technologies – that are used for implementing a QA System – and lastly constructing a Question Answering System based on a TKG. The constructed system tries to identify all the tourism entities in a question achieving that by using a Convolutional Neural Network (CNN) model and Attention mechanism amongst other methods. The experiments are carried out on the Xi'an tourism knowledge graph constructed by crawling, cleaning and storing data in the neo4j graph database [69].

Furthermore, Sui, Y. proposes a **Tourism KG Question Answering System** (TKGQA) that implements the **named entity recognition** (NER) model based on **Bert-BiLSTM-CRF** as well as the matching reasoning model based on **templates** [59]. Lastly, Do, P., Phan, T.H.V. and Gupta, B.B. present in detail how knowledge graphs as well as deep learning (natural language processing) can be used in order to develop a QA system for Vietnamese Tourism [20].

Chapter 4

Tourism KG Case Study: The Greek Island of Santorini

In **Chapter 4** we will **introduce and analyze the (Domain-Specific) Knowledge Graph** we created for the purposes of this thesis. As mentioned in earlier chapters, there are many different domains – namely some of them are media, publications, life sciences, law, tourism. We decided upon adopting the **Tourism** domain, mainly **because of the following reasons**:

1. Firstly, the reason we decided to choose this domain lies in the fact that, as mentioned earlier, Tourism as a domain offers a lot of **undiscovered opportunities** and it **plays a significant role in our society** – having contributed up to 20,8% of Greece's GDP in 2019 [73].
2. Secondly, because of the fact that **Tourism** is a domain **within our knowledge realm.**

As mentioned in Chapter 1, our **Tourism Domain Knowledge Graph is a part of the Question Answering System** we wish to build **for planning and answering all the questions someone might have regarding their next trip**, from what activities they can do to what attractions they can see to what important information they should be aware of before visiting. In order to achieve that, we chose **a specific – small in size – tourist destination** for our case study so as to create a **deep** and **rich** Knowledge Graph**.** As our case study we chose **Santorini** as it manages to combine all the said elements and also it is a popular destination for many people.

Through the next two sections we will refer to the procedure of creating the Tourism Knowledge Graph of Santorini. In **Section 4.1** we will analyze the self-constructed **domain-specific ontology** we created for the purposes of this Knowledge Graph on which our Question Answering System will be based. Whereas, in **Section 4.2** we will proceed to describe **how we built our KG** by referring to its population and creation method amongst other information regarding the KG building.

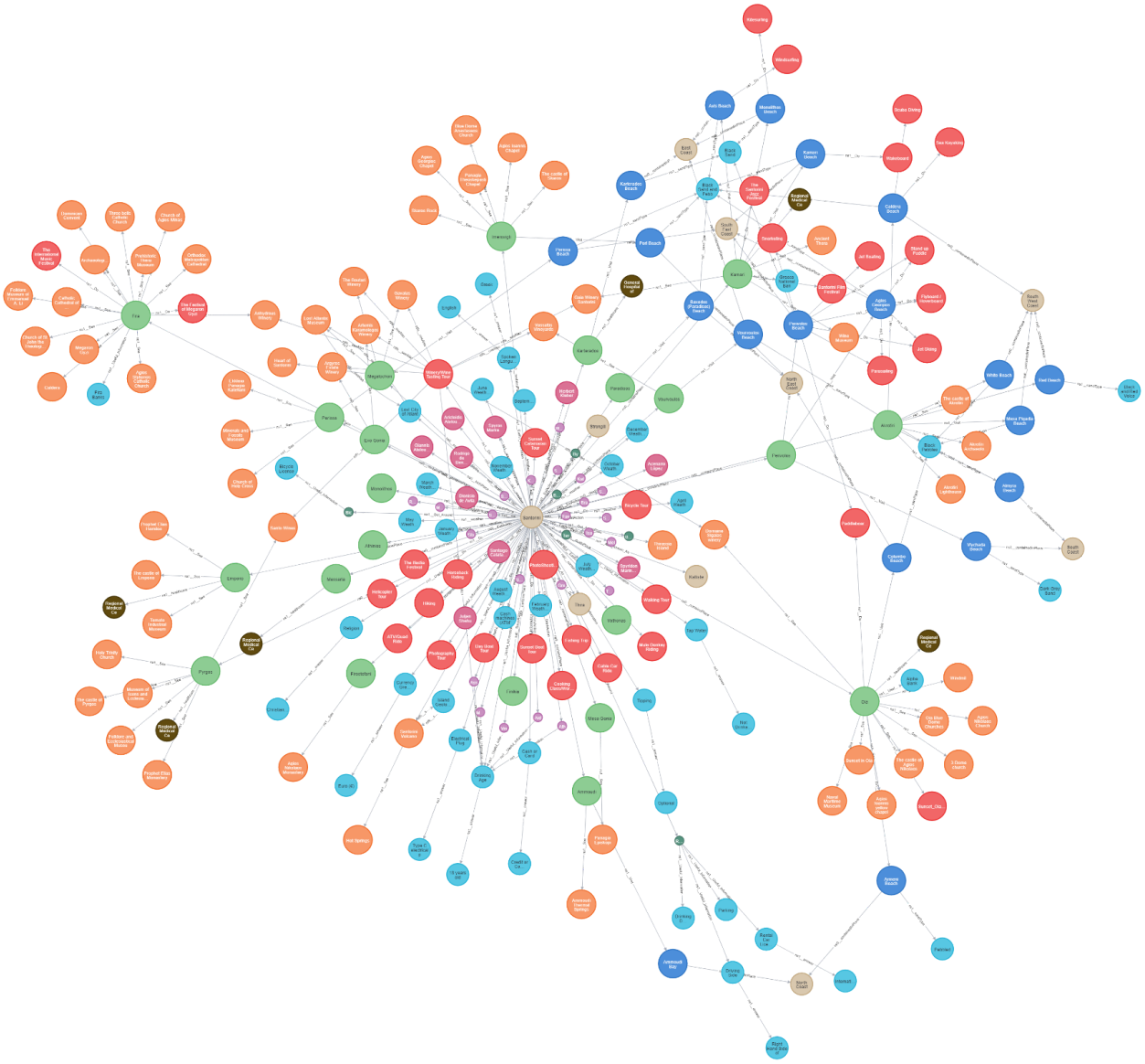

Figure 11. Snapshot of the complete Tourism Knowledge Graph of Santorini (237 nodes and 285 edges)

4.1 The Tourism KG Ontology

Through this Section we aim to describe the ontology classes and properties we created for the purposes of building a deep and rich in information **Tourism Knowledge Graph.** The inspiration behind creating such an ontology was to build an **organized and detailed Knowledge Graph** and a **better Question Answering System**.

In order to create that ontology we had to visualize what information we would like our KG to have. By researching as well as brainstorming what someone would want to know before visiting a new unfamiliar place we managed to create a **list of possible questions or question scenarios**.

Below we present **some question examples**:

1. *What can a newlywed couple do in Santorini?*
2. *Do I need to tip in Greece?*
3. *Family Friendly organized beaches in Santorini*
4. *What festivals take place in santorini?*
5. *People that were born in Santorini*
6. *Where can I find an ATM in Santorini?*
7. *Where can I best admire Santorini's sunset?*
8. *What papers do I need in order to rent a car in Santorini?*
9. *Are there any Hospitals in Santorini?*
10. *What is the weather like in Santorini during September?*

Some of the **key entities** that came up after filtering the list of questions are: **beaches** a tourist can visit, **tourist attractions** someone can see, **activities** someone can do, **useful information** that someone should know, amongst others. The above key entities inspired us into building our **Ontology Classes and Properties**. In order to properly represent the **diverse knowledge** and build a very **deep and rich semantic network** we have created an ontology of **52 classes** and **17 properties**.

In **Subsection 4.1.1** we will describe the **ontology classes** we have created and will help us better **describe** our Knowledge Graph entities. In **Subsection 4.1.2** we will describe the **properties** we have created in order to better **interlink** our KG entities and add more knowledge to the graph. The ontology analyzed in this section can be found (as an RDF file) here.

4.1.1 The Classes and Subclasses of the Ontology

First and foremost in order to describe a geographical location as well as the various villages that Santorini has we have created the superclass **Place** and the subclass **Village**.

Furthermore, as expected an island has many beaches and in order to describe that we have created the superclass **Beach** and its subclasses: **Well Organized**, **Partly Organized** and **Non-Organized**. We must notice the fact that a beach can belong to only one type of organization so it can either be well, partly or non organized.

We have also created appropriate superclasses (**Tourist Attraction**) and subclasses (**Natural Attraction**, **Archaeological Site**, **Castle**, **Museum**, **Place of Worship**, **Winery**, **Landmark**) for the description of the many tourist attractions a Greek tourist destination such as Santorini may have. A natural attraction may describe a Volcano, hot springs and other naturally made attractions. Meanwhile, the class place of worship can describe a orthodox or catholic church (or cathedral).

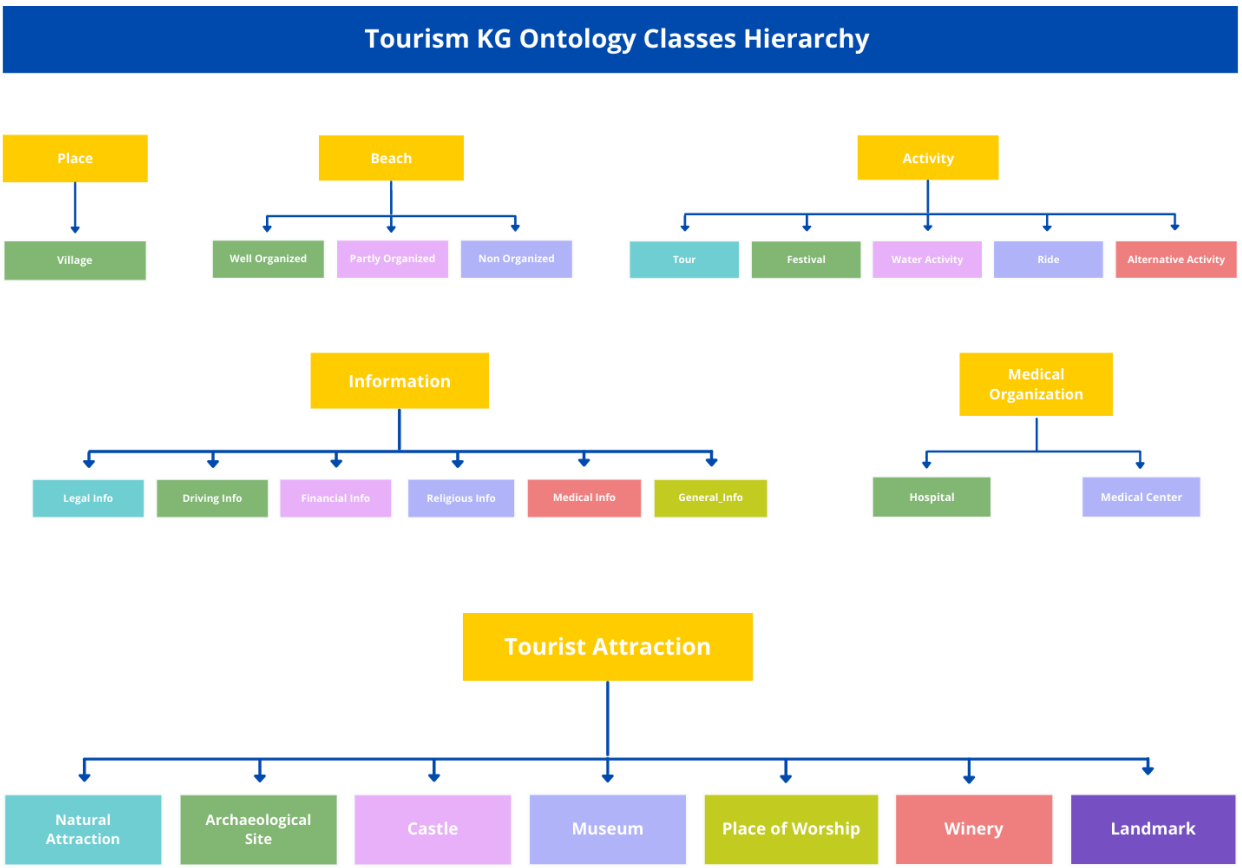

Figure 12. Classes Hierarchy of our self-constructed Tourism KG Ontology

Additionally, in order to describe the things, activities, a tourist can do we have created a superclass named **Activity** and the subclasses **Tour**, **Festival**, **Water Activity** (referring to the activities someone can do on or besides the water/sea), Ride and Alternative Activity (in order to describe any activity that is not as famous or usual).

Moreover, we have the superclass **Information** in order to describe any useful information someone should know beforehand. The above superclass has been broken down into 6 subclasses in order to better describe the diversity of information; namely we have: **Legal Info**, **Driving Info**, **Financial Info**, **Religious Info**, **Medical Info**, **General Info**. We have created the superclass, Medical Organization, and its subclasses, Hospital and Medical Center, in order to describe the medical facilities a place (Santorini) has and may interest the tourist.

We have created some additional classes (**Sand**, **Local Cuisine**, **Local Wine**, **Transportation**, **Notable Person**, **Climate**, **Myth**) that help us, in addition to the others, to better describe the entities on our KG. Lastly, we have created some more classes that assist us in describing our data and create a more deep and rich Knowledge Graph. Those classes are: **Romantic Place, Family Friendly Place, Picturesque, Romantic Activity, Famous, Family Friendly Beach, Touristy, Long, Romantic Beach, Small, Quiet, Upscale, Secluded, Nudism Friendly, Romantic Attraction.** In the Table below we are able to overview the ontology classes, subclasses and descriptions.

Name	Subclass of
Village	Place
Myth	-
Local Cuisine	-
Local Wine	-
Transportation	-
Notable Person	-
Sand	-
Climate	-
Activity	-
Tour	Activity
Festival	Activity
Water Activity	Activity
Ride	Activity
Alternative Activity	Activity
Beach	-
Well Organized	Beach
Partly Organized	Beach
Non Organized	Beach
Tourist Attraction	-
Natural Attraction	Tourist Attraction
Archaeological_Site	Tourist Attraction
Castle	Tourist Attraction
Museum	Tourist Attraction
Place of Worship	Tourist Attraction

Name	Subclass of
Winery	Tourist Attraction
Landmark	Tourist Attraction
Medical Organization	-
Hospital	Medical Organization
Medical Center	Medical Organization
Information	-
Legal Info	Information
Driving Info	Information
Financial Info	Information
Religious Info	Information
Medical Info	Information
General Info	Information

Table 4. Overview of the Ontology **Classes**

4.1.2 The Properties of the Ontology

The properties we have created in order to interconnect our data in a better, more descriptive way can be seen with more detail in **Table 5** below. First of all, **Visit** is a property that we use when we want to connect a Beach with the Place, more specifically the Village, that it is located in and that someone can visit. Furthermore, we have got properties like **Sand Type** in order to connect a Beach with the type of its sand (Black Sand, Pebbles etc.) as well as **Blue Flag** in order to refer to the fact that a Beach has or has not a Blue Flag.

Secondly, we have created the property **Do** to connect an Activity to a Place (Village) or a Beach. Additionally, we use the property **during** in order to connect a Festival with the month in which it takes place. Moreover, in order to interconnect the various Tourist Attractions that Santorini has and the Places (Villages) that they are located into and that a tourist can see and admire, we have created the property **See**.

With the property **Useful Information** and **answer** we aim to refer to some additional information that is of important use to the tourist and may regard any entity. Furthermore, we have created properties like: **Also Known As, Brief History**, **Covid Protocols**, **Get Around**, **Island Formation,**

Myths and Name Origin in order to add more information about a Place's *alternative names, history, latest covid related protocols, different means of transportation, geological formation, myths* and *name origin* respectively.

In addition, we have created the property **Healthcare** in order to connect the various medical organizations – meaning the Hospitals and Medical Centers – with the Place (Village) that they are located in. Lastly, we have got the property **Weather** which helps us connect any Place (more specifically Santorini) with its climate.

Name	Domain	Range
Also Known As	Place	Place
answer	Information	-
Blue Flag	Beach	-
Brief History	Place	-
Covid Protocols	Place	-
Do	Place, Beach	Activity
during	Festival	-
Get Around	Place	Transportation
Healthcare	Place	Medical Organization
Island Formation	Place	-
Myths	Place	Myth
Name Origin	Place	-
Sand Type	Beach	Sand
See	Place	Tourist Attraction
Useful Information	*(Any Class)*	Information
Visit	Place	Beach
weather	Place	Climate

Table 5. Overview of the Ontology **Properties**

4.2 The Creation and Population of the Knowledge Graph of Santorini

In this Section we will focus on the creation and population of the Knowledge Graph of Santorini. We have successfully created a (domain-specific) Knowledge Graph focusing on the Tourism of the island of Santorini consisting of **237 nodes and 285 edges.** We must point out the fact that we have used the **Turtle RDF Language** in order to serialize our KG. For more information on **RDF Terminology and KG Creation and Hosting** see **Chapter 2, subsection 2.1.2.**

Moreover, we must notice the fact that we have hosted the Tourism KG that we have created on the **Neo4j Graph Data Platform**. In order to enable the use of RDF in Neo4j we have made use of the **Neosemantics (n10s) Plugin**. For more information about the Neosemantics plugin see here. Additionally, for the purposes of **populating** our **Tourism Knowledge Graph** we have collected both **structured** and **unstructured data** related to Santorinian Tourism. It is important to notice the fact our Knowledge Graph contains mainly **static data.**

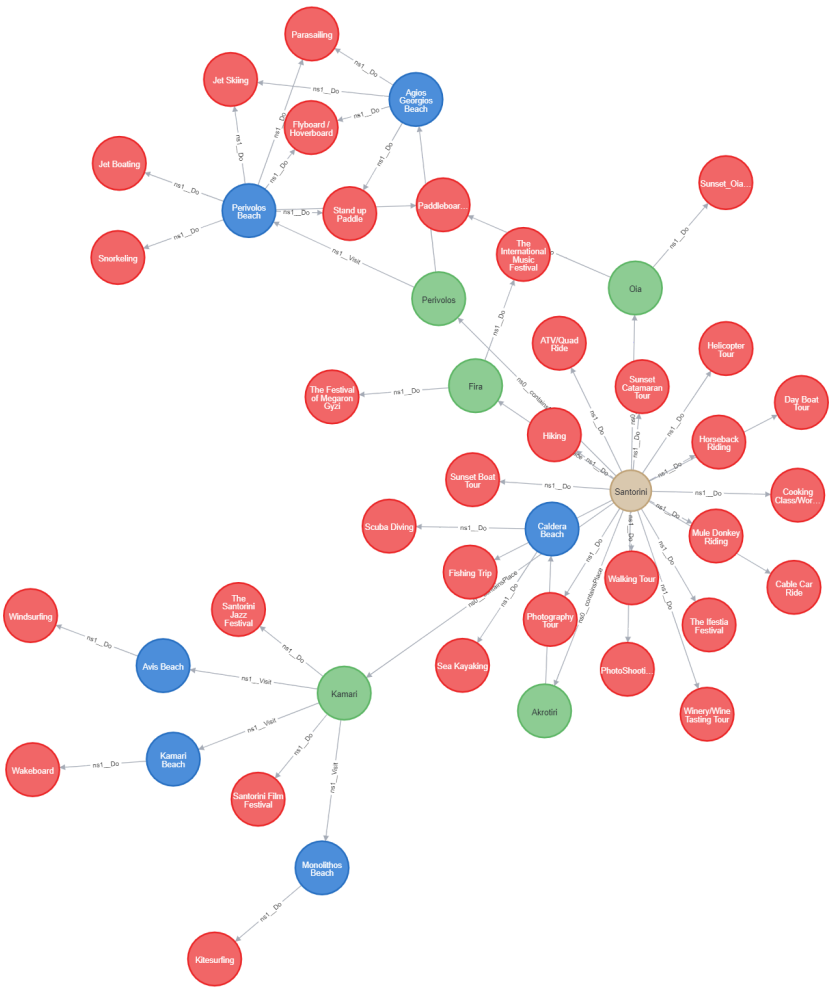

Figure 13. An example of the **Tourism Knowledge Graph** of Santorini (all nodes with **label:Activity**)

For the population of this KG we managed to collect and "clean" a variety of data deriving from different sources. We collected any **structured data** from the respective **DBpedia** entry of Santorini [4].
The key **unstructured data sources** of this project are as follows:

1. Firstly we consulted **Destination Management Organizations (DMOs)** [54] and popular **Travel Blogs** [33] [34] [2] [29] [56] [1] [55].
2. Secondly, we turned to **Geographical Information Systems (GIS)** – mainly **Google Maps** [28].
3. Additionally, we have used information about Santorini taken from both Wikipedia [52] and Wikitravel [53].
4. Lastly, information not yet covered by the already mentioned data sources was collected from vrisko.gr [74] and tripadvisor.com [60].

Below we are going to take a look at an example of our **Turtle RDF file** that is available to study in more detail **here**.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix sch: <https://schema.org/> .
@prefix stkg: <http://santorinitourismkg.com/vocab/> .
@prefix dbp:    <http://dbpedia.org/property/> .

@prefix obj: <http://SantoriniTourismKG/genObj#> .
@prefix places: <http://SantoriniTourismKG/Places#> .
@prefix people: <http://SantoriniTourismKG/People#> .
@prefix attr: <http://SantoriniTourismKG/Attractions#> .
@prefix activ: <http://SantoriniTourismKG/Activities#> .

#Villages

places:Akrotiri sch:name "Akrotiri" ;
      a stkg:Romantic_Place ;

      stkg:Visit places:Red_Beach, places:MesaPigadia_Beach,
      places:White_Beach, places:Caldera_Beach, places:Almyra_Beach;

      stkg:See attr:Akrotíri_Archaeological_Site, attr:Akrotiri_Castle,
      attr:Akrotiri_Lighthouse;

      a stkg:Village .

places:Ammoudi sch:name "Ammoudi" ;
      stkg:Visit places:Ammoudi_Bay ;

      stkg:See attr:Ammoudi_ThermalSprings;
```

```
      a stkg:Village .

places:Athinios sch:name "Athinios" ;
      a stkg:Village .

places:Emporio sch:name "Emporio" ;

      stkg:healthcare places:Emporio_RMC ;

      stkg:See attr:Emporio_Castle, attr:TomatoIndustrial_M,
      attr:ProphetElias_Hamilos;

      a stkg:Village .


places:Fira sch:name "Fira" ;

      stkg:Do activ:The_Festival_of_Megaron_Gyzi,
      activ:The_International_Music_Festival;

      stkg:See attr:Caldera, attr:Folklore_M,
      attr:ArchaeologicalMuseum_Thera, attr:PrehistoricThera_M,
      attr:Megaron_Gyzi, attr:Cathedral_StJohntheBaptist,
      attr:StJohnTheologian, attr:AgiosMinas,
      attr:OrthodoxMetropolitanCathedral, attr:ThreeBells,
      attr:Dominican_Convent, attr:AgiosStylianos,
      attr:Anhydrous_Winery;

      stkg:Useful_Information obj:Fira_Banks;

      a stkg:Village .
```

Example 2: Code Snippet of the Knowledge Graph Turtle RDF file.

Above we can see a small example of our Turtle RDF file containing our Tourism Knowledge Graph about Santorini. The above code snippet starts by showing the **PREFIX declarations** and then proceeds by showing some example **triples** portraying some of the **villages** of Santorini and the **rich information** that is linked to them.

Firstly, regarding the **PREFIX declarations** at the beginning of the example, as we have mentioned in earlier chapters (see Chapter 2, subsection 2.1.2.) the **Turtle RDF** format allows us to use **namespace prefixes.** A **PREFIX** is a way to **shorten a long namespace** (IRI) into a shorter local name and that's exactly what we have done in our code. We have created prefixes such as sch in order to shorten the namespace (IRI) https://schema.org/. The same process has been followed for the rest of the namespaces. We have created prefixes for both the common RDF Vocabularies that we use in our code and for the ones created for the purposes of this Knowledge Graph.

Secondly, regarding the RDF triples that are shown in the code snippet above, as we have already mentioned in Chapter 2 an RDF triple consists of a two nodes – the source node (**Subject)** and the target node (**Object)** – and a triple connecting them (**Predicate**). In our example, a triple would be places:Akrotiri **sch:name** "Akrotiri" or places:Akrotiri **stkg:Visit** places:Red_Beach. In both examples, the **Subject** is the **village "Akrotiri"** (places:Akrotiri) but in the first example the **Object** is the "Akrotiri" – which is a *Literal* – whereas in the second example the **Object** is the **beach "Red Beach"** (places:Red_Beach). In addition, in the first example, the **Predicate** is the **sch:name** whereas in the second example the **Predicate** is the **stkg:Visit** .

It is true that the **Turtle RDF** Language offers us a lot of assisting tools like the **Prefixes** we mentioned earlier but also it offers us the ability to add **Multiple Predicates**, **Multiple Objects** and **Types** and that is what we have done in the code snippet above**.** Below we can see in coloured detail where we have used the multiple predicates, objects and types.

```
places:Akrotiri sch:name "Akrotiri" ;
      a stkg:Romantic_Place ;
      stkg:Visit places:Red_Beach, places:MesaPigadia_Beach,
      places:White_Beach, places:Caldera_Beach, places:Almyra_Beach;
      stkg:See attr:Akrotíri_Archaeological_Site, attr:Akrotiri_Castle,
      attr:Akrotiri_Lighthouse;
      a stkg:Village .
```

More specifically, in the code snippet above we can notice that the **village "Akrotiri"** is **described as a Village and as Romantic Place**. It **has the name Akrotiri** and someone can **visit various beaches** in Akrotiri like the **Red Beach, the Mesa Pigadia Beach, the White Beach** etc.. Furthermore, if someone is in the village "Akrotiri", they can **see various attractions** like the Akrotiri Archaeological Site, the Akrotiri Castle, etc.

Figure 14. Graph Representation of the Village "Akrotiri" and its directly connected entities.

Chapter 5

Behind the Question Answering System

In Chapter 5, we will be doing a thorough presentation and analysis of our Question Answering System. This chapter aims to answer all the questions regarding the nature of our QA system, the building of its subsystems and how it is able to implement the Tourism domain Knowledge Graph we have created.

In the next 5 Sections we are going to be analyzing each of the Question Answering subsystems. At first, in Section 5.1, we are going to be showing an overview of our system as a whole. In section 5.2 we will go into detail about the Template-Based model and the Template Library. In Section 5.3 we will start analyzing the first significant subsystem of our QA system called Question Classification. Furthermore, in Section 5.4 we will present the second important subsystem regarding Template Matching. Lastly, in section 5.5 we will talk about the last but not least subsystem called Answer Retrieval. We have to highlight the fact that in order to build our QA system we have implemented the python language and we have used the Google collaboratory tool to write our python script. Here you can find the colab notebooks for the **Question Classification** and the **Template Matching** as well as a complete version of our Question Answering system. Additionally you can find them in this repository here.

5.1 Question Answering System Overview

As stated in Chapter 2, Section 2.2, Question Answering Systems are able to automatically answer a question asked by a human [64]. and they can be divided into different categories. Our QA System can be characterized as a closed domain question answering system as it aims to answer questions under the specific domain of Tourism. Additionally, our system deploys a Template-Based Architecture in order to answer the factoid questions posed by the user. For more information on that you can check out Section 5.2.

In the flowchart below we are able to see an overview of the complete QA system we have built as well as how our system flows from receiving our user's question to retrieving its answer according to the template we have matched it to.

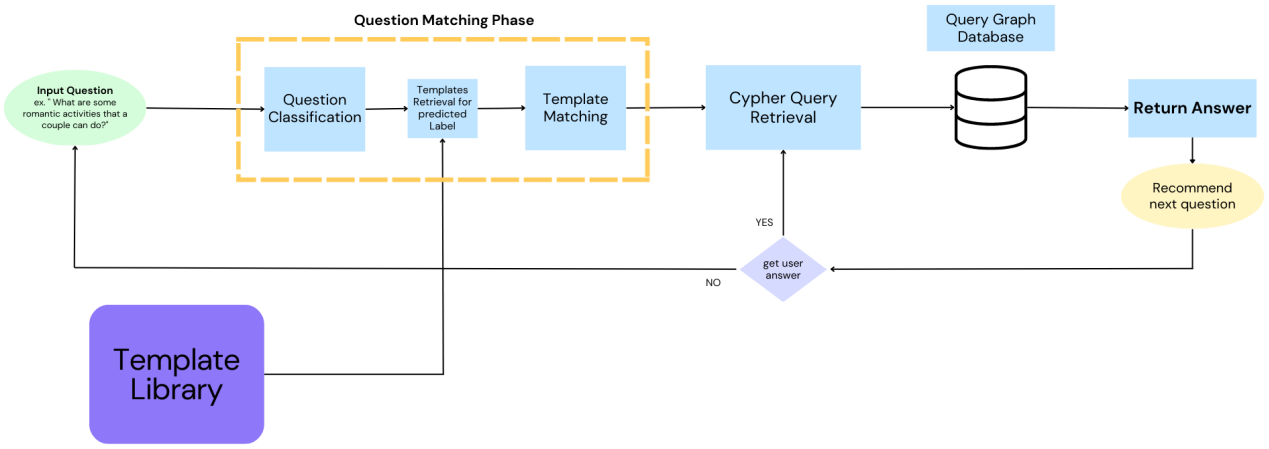

Figure 15. Flowchart of the complete Question Answering System we have built

As we can see in Figure 15. the flow our system follows starts with an input Question provided by the user in natural language (English) and then proceeds to go through the rest of the system consisting of the Question Matching Phase and the Answer Retrieval Phase. In the meanwhile, our QA system is being linked to the handcrafted Template Library we have established based on our Knowledge Graph. Our system can be divided into **3 main parts** or subsystems:

1. **The Question Classification Subsystem**
   Key points:
   - ➔ No Training Set
   - ➔ Word Clusters
   - ➔ BERT Embeddings
2. **The Template Matching Subsystem**
   Key points:
   - ➔ Two Methods
   - ➔ SentenceTransformers
   - ➔ Cosine Similarity
3. **The Answer Retrieval Subsystem**
   Key points:
   - ➔ Cypher Queries
   - ➔ NER
   - ➔ Next Question Recommendation

Arguably, there are many question answering systems that are relying on a provided document(s) in order to base their search and retrieval. Alternatively, we propose a Question Answering System based on a Knowledge Graph (KGQA) that offers the storing of the knowledge in a rich semantic way and thus the retrieving answer to be more precise and structured. It is true that this kind of technology is gradually being adopted and gaining importance even in the business environment.

Furthermore, regarding our QA System as we mentioned earlier we were very attentive to its building since we created our Tourism Knowledge Graph and its ontology. As we are going to see

in the next few sections our KG ontology has played an important role in both creating the Templates for our Template Library (Section 5.2) as well as coming up with our classes in Question Classification (Section 5.3).

5.2 Template Library

In this Section we are going to present the Template Library we have established for our Template -Based Question Answering System. We are going to analyze the handcrafted templates we have created; what are they, why do we need them and of course how did we generate them.

First and foremost, the Template-Based model is used for answering closed domain factoid questions by matching the question to the **template(s)** stored in a "library". Afterwards the best fitting template is chosen and the appropriate query is retrieved [72]. The Template Library is a set of hand-crafted templates. A template is a pair of **natural language pseudo-question** and a **query** with some slots acting as **entity placeholders** (< >). Inside those templates reside patterns that can be either handcrafted or automatically generated [72].

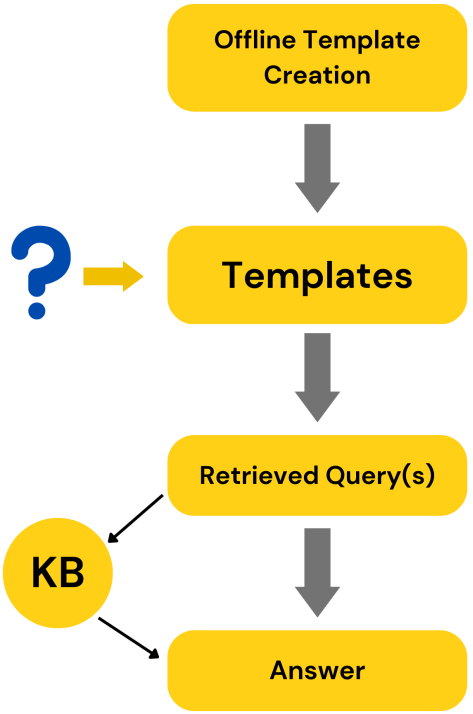

Figure 16. Illustration showing a Template-Based Architecture

Furthermore, we have to highlight the fact that templates play a significant role in question answering over knowledge graphs (KGQA), considering the fact that they create a bridge between the input question and the Knowledge Graph. However, it is true that the Template-Based model

can have its disadvantages. This architecture can turn out to have limited capabilities as well as limited scalability since they are fixed and often created manually by experts meaning that it could require a lot of human effort, time and also maintenance.

We have manually defined our templates that you can see in Table 6. The table below shows a few example pairs of natural language questions and Cypher queries.

Natural Language Question	Cypher Query
What are some romantic places?	MATCH (start{ns0__name:"Santorini"})-[r:ns0__containsPlace]->(end:ns1__Romantic_Place) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
What do you know about <Place>?	MATCH p=(start{ns0__name:"<Place>"})-[r]->(end) RETURN start.ns0__name AS Entity, labels(start) AS Labels, Type(r) AS Relation, end LIMIT 10
Villages in Santorini	MATCH (start{ns0__name:"Santorini"})-[r:ns0__containsPlace]->(end:ns1__Village) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end AS Entity2
Family-Friendly Beaches in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(end:ns1__FamilyFriendly_Beach) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end AS Entity2
Are there any well-organized beaches in <Place> and what can I do there?	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(m:ns1__Well_Organized)-[r:ns1_ _Do]->(end) RETURN start.ns0__name AS Entity, m AS Beach , Type(r) AS Relation, end.ns0__name AS Entity2_Name
Is <Festival> taking place during <Month>	MATCH (p) WHERE p.ns0__name="<Festival>" RETURN p.ns0__name, p.ns1__during;
Is <Notable Person> born in Santorini?	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__birthPlace]->(end:ns1__Notable_Person) WHERE end.ns0__name="<Notable Person>" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name, end.ns0__jobTitle AS Entity_JobTitle
Where can I find a hospital or medical center?	MATCH p=(start)-[r:ns1__healthcare]->(end) WHERE end:ns1__Medical_Center OR end:ns0__Hospital RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name

Table 6. Example templates (natural language, cypher query) from our template library (for more see Appendix A.)

Regarding the handcrafted Templates we have created (and can be seen in the Table above) we can notice that they are mainly based on the Ontology of our Knowledge Graph that we have described in full detail in Chapter 4, Section 4.1. We were heavily inspired by the types of entities and generally the rich information stored in the graph. Our goal when designing the Templates was to be able to capture the interlinked entities in the TKG as well as simplify the "translation" between the structured Knowledge Graph and the unstructured natural language questions. Moreover, regarding our cypher queries, we can notice the fact that we have decided to retrieve as much interconnected information and knowledge as possible. As we can see, our queries instead of answering shortly to what the user asks, they follow a path and they return a more complete and extensive answer. We think that in a domain such as Tourism and a Question Answering system with a goal such as ours the user would be more interested and keen on absorbing and discovering as much knowledge about their tourist destination as possible.

5.3 Question Classification

In Section 5.3 we will be **presenting and analyzing the algorithm behind** one of the 3 most important subsystems of our Question Answering System that we already mentioned above, the **Question Classification** subsystem. We must notice the fact that we were heavily inspired by the "BERT for Text Classification with NO model training" tutorial you can find here. The **Question Classification** subsystem is the first stage – out of the two stages of the Question Matching phase – that the input question must go through in order to get classified in the right group of Templates.

In this study, for the purposes of building our Question Answering System, we have used the state-of-the-art NLP framework, **BERT** which stands for **Bidirectional Encoder Representations from Transformers**. **Google's BERT is a transformer-based architecture** originally released by Google in 2016 in the following paper: "Attention Is All You Need" [61].. In 2019 the BERT model proposed by Jacob et al. got published in the paper called "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [18].

We have to highlight the fact that BERT's key feature is that it applies Attention mechanisms instead of RNN in order to collect the contextual information of a given word and then encode that in a rich semantic vector that fully represents the word. In addition, we must notice the fact that Google's BERT has undoubtedly inspired many other recent NLP architectures.

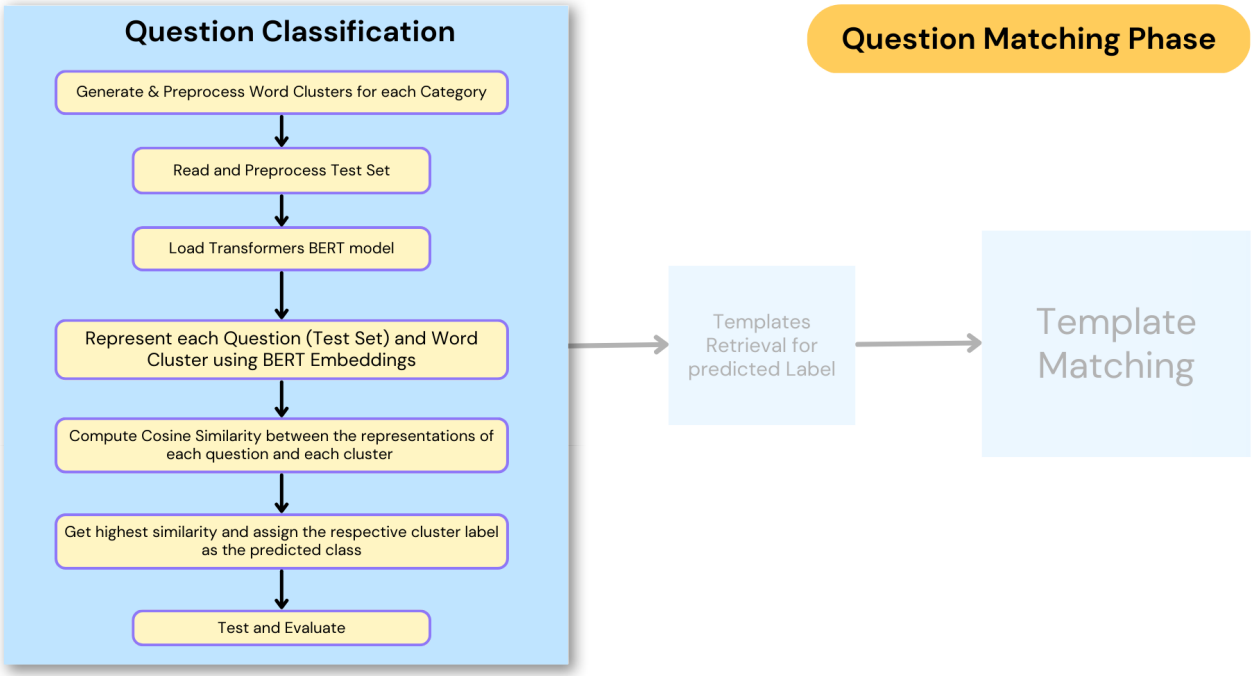

Figure 17. Flowchart of the Question Classification subsystem

The **complete Question Classification subsystem algorithm** that we are going to be analyzing shortly, along with the test set we have created – for the purpose of conducting our experiments (*for more see Chapter 6*) – can be found in the repository here. Additionally, you can find the Google Colab Notebook containing our python script here.

We started creating our **Question Classification** subsystem by importing all the necessary packages that we are going to be needing. After that we focused on creating our Target Word Clusters, one for each category. An important fact that we should highlight is the lack of data sets regarding the Tourism domain that we could use to train our model. This limitation led us to employ a different approach.

As a result we have used the gensim library to create keywords that will represent the context of each of our categories. As we can see we have created a method called get_similar_words that gets a few keywords, the number of most similar words that we want and of course our pre-trained Word Embedding model we have previously loaded and it returns a list of n most similar words. You can read more about the gensim library here.

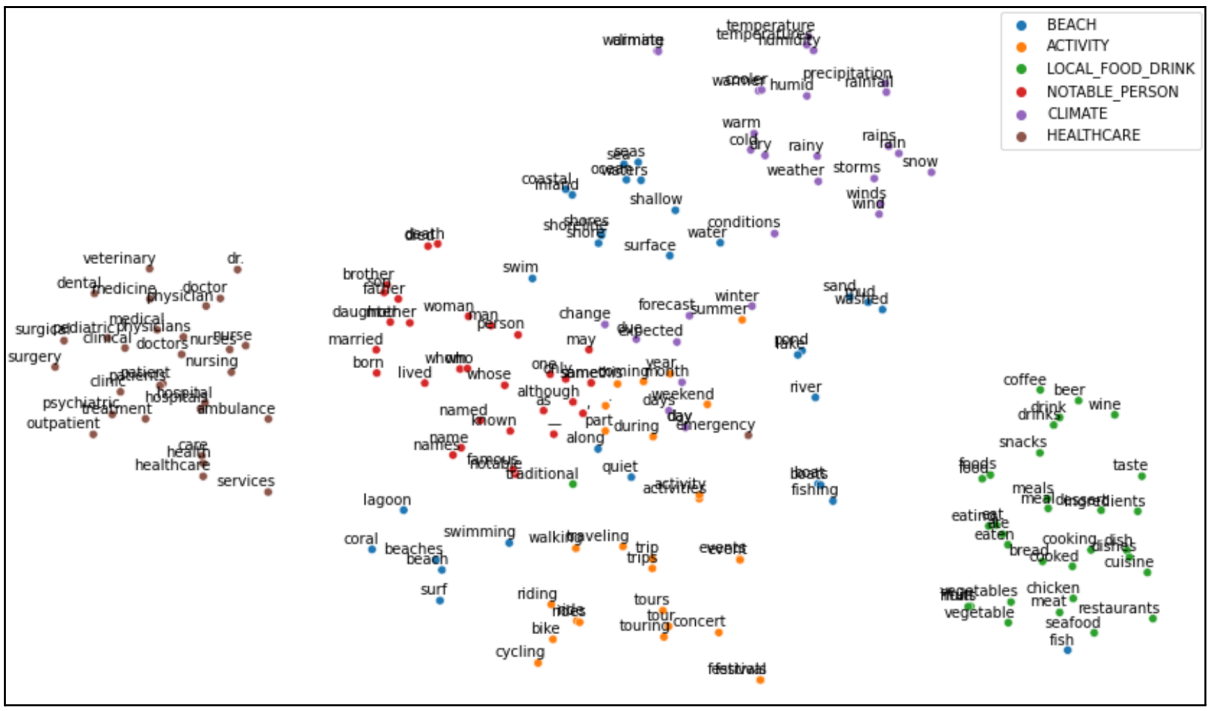

Figure 18. Scatter plot presenting some of our word clusters

Furthermore, after creating our word clusters we created two preprocessing methods called Cleaning_1_1 and Cleaning_1_2 for preprocessing our auto-generated word clusters. Their main purpose is to iterate through the list of words and remove words, punctuation marks and null entries that offer zero context and to remove any repeated words between the clusters. Afterwards we use those preprocessing methods to conduct our experiments (*for more see Chapter 6*). Our small Test Set containing Tourism Questions for our use case of Santorini is contained in a csv file. We read it and transformed it to a pandas dataframe and after that we applied a preprocessing method to our dataset in order to clean our data; cleaning text, removing stop words, and applying lemmatization.

Our next step is to load the original pre-trained version of BERT using the package transformers. We have used the bert-base-uncased BERT model for our Question Classification subsystem. Read more about the Transformers BERT model here. Our purpose is to implement the state of the art technology of BERT Word Embeddings in order to represent our questions with an array (shape: n words x 768) and then summarize each question into a mean vector. By using the method BERT Embeddings we manage to represent our questions with a 768-dimensional vector. After creating our question representations and storing them in a numpy array, we do the same with our keywords in the target word clusters we have created earlier. We are going to create a dictionary containing the label and the respective mean vector of each word cluster.

Finally, after creating our representations of our Test Set and our Target Labels we can start building our model for question classification. The main idea behind our model is to compare each question to each target word cluster by computing the similarity between the two vectors. We have decided upon using the Cosine Similarity implementation of scikit-learn that returns an array of similarity scores with shape: number of questions x number of labels (9: PLACE, BEACH, ACTIVITY,

TOURIST_ATTRACTION, LOCAL_FOOD_DRINK, NOTABLE_PERSON, CLIMATE, HEALTHCARE, INFORMATION).

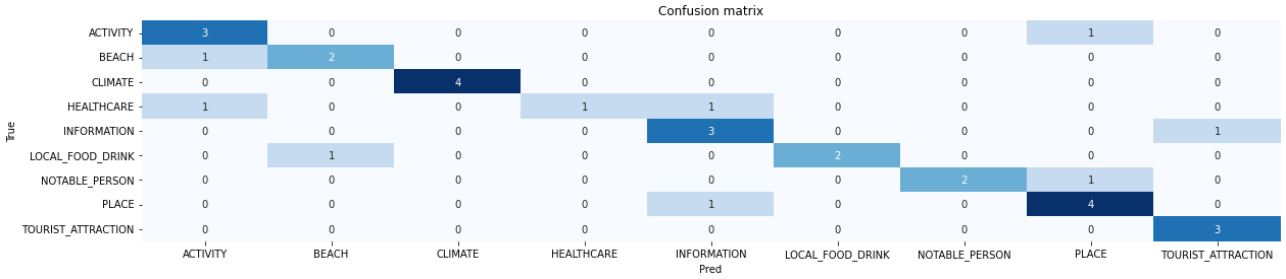

Figure 19. Confusion matrix of the Question Classification subsystem

After that we are going to choose the highest similarity for each question and assign that label as the predicted class. After that follows the testing and evaluation part where we have used the following evaluation metrics: Accuracy, AUC, Precision, Recall, F1-Score and of course we have created confusion matrices. For more information about our experiments and results see Chapter 6. Ultimately we have achieved an accuracy score of 0.75.

5.4 Template Matching

In Section 5.4 we are going to be describing our second main subsystem and the second stage of the Question Matching Phase that is the Template Matching Subsystem. When designing this subsystem the main idea was that after sorting the input question into the best fitting group of templates we needed to compare that question with each template. In order to do that we wanted to submit the question to a number of similarity tests and then compare and combine them in order to achieve a better accuracy in our system. Here you will find the repository containing the **complete Template Matching subsystem algorithm** along with the test set and test templates we have created for the purposes of conducting our experiments (see Chapter 6). Additionally, you can find the Google Colab Notebook containing our python script here.

The Template Matching Subsystem consists of two subsystems:

- The **Sentence Similarity (SS)** Subsystem based on SentenceTransformers
- The **Linguistic Similarity (LS)** based on a more linguistic approach

Below we can see a detailed flowchart of our complete Template Matching Subsystem, with each Similarity method and its compartments.

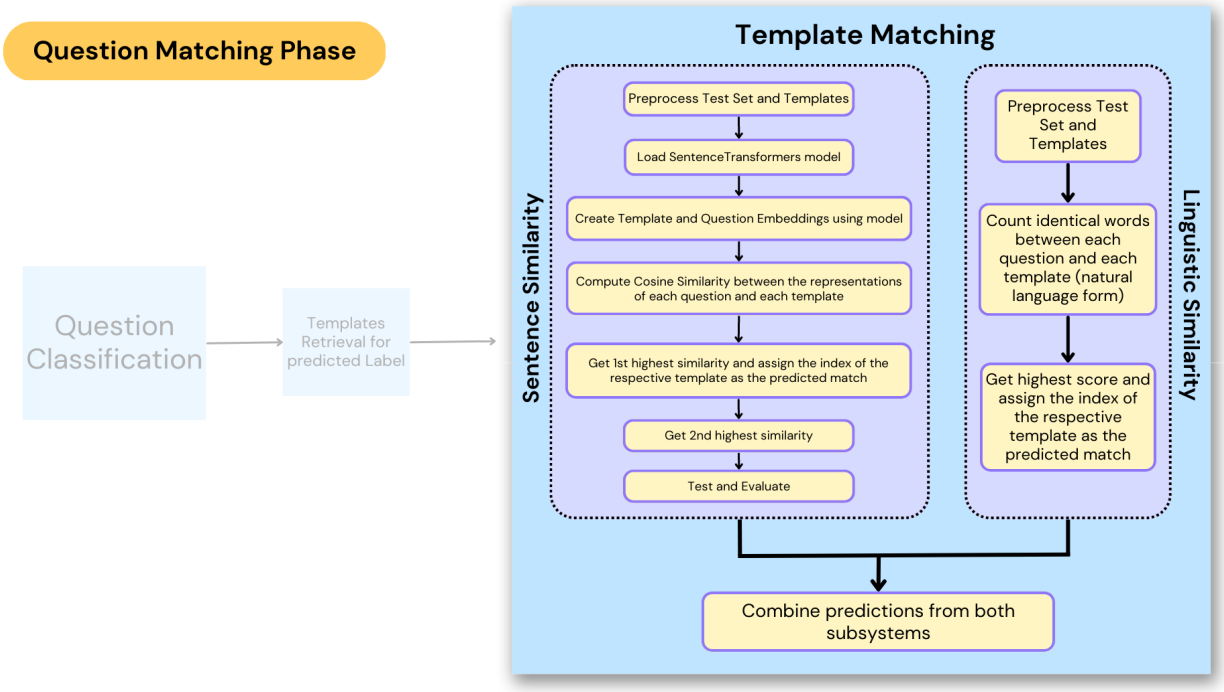

Figure 20. Flowchart of the Template Matching subsystem

At first, after importing the necessary packages we read the csv files for the test templates and small Test Set and then we transform them to a pandas dataframe. After that we move on to implementing our two subsystems mentioned previously. When we have each prediction, meaning the predicted template that our question fits better to, we compare them to each other and based on the algorithm shown below (Algorithm 1) we manage to get the final prediction.

Algorithm 1 *SubsystemsPredictionsComparison(PredSS, PredLS, SecPredSS)*

Input: The 1st predicted template *PredSS* from the Sentence Similarity method,
The predicted template *PredLS* from the Linguistic Similarity method,
The 2nd predicted template *SecPredSS* from the Sentence Similarity method;
Output: The final predicted template *final_Pred.*

```
if PredLS != NULL then
    if PredLS == PredSS then
        final_Pred ← PredSS
    else if PredLS == SecPredSS then
        final_Pred ← SecPredSS
    else
        final_Pred ← PredLS
else
    final_Pred ← PredSS
return final_Pred
```

Firstly, we check if the LS method's prediction exists and then we check if it is equal to the Sentence Similarity's prediction then we set the final prediction equal to that. Alternatively, if the PredLS is equal to SS second predicted template then we set the final prediction equal to the

SecPredSS. If none of the above is true then we set the final_Pred equal to the LS method's prediction. However, if we find out that the PredLS does not exist we set the final prediction equal to *PredSS.*

Moreover, we must highlight the fact that in order to calculate the similarity between the question and the templates of our group we have used the straightforward brute force method of exhausting every q,t pair and computing the similarity between the question q and the natural language part of each template t. In the next two subsections we are going to be analyzing each of our subsystems, starting with the Sentence Similarity Subsystem in SubSection 5.4.1 and following with the Linguistic Similarity one, in SubSection 5.4.2 .

5.4.1 Sentence Similarity using SentenceTransformers

In subsection 5.4.1, we will be presenting and recording each step we took towards the creation of our Sentence Similarity Subsystem. We started by preprocessing our data (Test Set and Templates) with the two methods that we have created for that purpose. In order to calculate the similarity between the question and the templates we have implemented a pre-trained SentenceTransformers model that will allow us to transform our q and t into vectors that manage to capture the semantic information. According to the official documentation of SentenceTransformers:

*"**SentenceTransformers** is a Python **framework** for state-of-the-art sentence, text and image embeddings. The initial work is described in our paper **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**."* [51]

This framework can be used in order to compute sentence embeddings and compare them to find sentences with similar meaning. It is true that this kind of technology can be used to achieve many tasks like: semantic textual similarity, semantic search, or paraphrase mining. Moreover, this framework lays its foundation on PyTorch and Transformers and offers a large collection of pre-trained models [58]. After conducting a plethora of experiments (*see Chapter 6*) we have decided to use the "all-MiniLM-L12-v2" SentenceTransformers model. However in Table 7, you can see an overview of the other models we considered as well.

	Base Model	Dimensions	Training Data	Size
all-mpnet-base-v2	microsoft/mpnet-base	768	1B+	420 MB
all-distilroberta-v1	distilroberta-base	768	1B+	290 MB
all-MiniLM-L12-v2	microsoft/MiniLM-L12-H384-uncased	384	1B+	120 MB
all-MiniLM-L6-v2	nreimers/MiniLM-L6-H384-uncased	384	1B+	80 MB

Table 7. Overview of the SentenceTransformers pre-trained models [58]

Finally, after creating our vectors we move on to comparing each question vector to each template vector in order to compute how similar they are to each other by using the Cosine Similarity. As a

result we get a list of similarity scores representing how similar (semantically) each question is to each template.

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 21. Cosine Similarity definition [63].

Afterwards, for each question in our Test Set we get the 1st higher similarity score as well as the 2nd highest similarity score and store it for later use. Finally, as our final prediction we assign the "T" + index of the corresponding template with the 1st higher similarity score. We then proceed with testing and evaluating our subsystem.

5.4.2 Linguistic Similarity Method

In this subsection we will describe the system behind the second subsystem of our Template Matching process. Firstly, we similarly start by preprocessing our Test Set and test templates with the preprocessing methods mentioned earlier in Subsection 5.4.1 but in this case we remove any stopwords, too. Afterwards we create and deploy the **Similarity_LinguisticApproach** method for computing the **Linguistic Similarities of each question and each template**. The main function of this method is to count the identical words between the cleaned question and each cleaned template (Natural Language pseudo-question). As a result we get a list of lists. This list has (number of questions) lists and each list has (number of templates) elements. After that we find the highest score in each list and then we assign the "T" + index of the corresponding template as this method's final prediction.

5.5 Answer Retrieval

In this section we are going to be analyzing the last subsystem of our Question Answering system that aims to connect the Knowledge Graph we created with the QA system. The script we are going to be analyzing below can be found in the same colab notebook as the Template Matching, here.

As we can see in figure 15, after getting our user's question and having gone through the whole Question Matching phase successfully, meaning that we have matched our input question to a template, then we are ready for the **Answer Retrieval phase**. In the next phase we are going to be "translating" the user's question into a language that our knowledge graph database understands. As we already mentioned earlier in Chapter 4, Section 4.2, we have hosted our KG into neo4j, therefore the language we need to translate it into is Cypher. If we learn how to communicate with the KG

database, we can finally question it about any tourism related information we would like to know and retrieve the respective answer. To do that we have created our Template Library, that we have presented previously in Section 5.2, consisting of two parts; a natural language pseudo-question and a cypher query.

First and foremost, we had to preprocess our cypher queries in order to find any entity placeholders (< >) that might exist and replace them with the special character "#". After preprocessing and connecting to neo4j using the py2neo driver we design two methods; the **answer_Retrieval method** in charge of returning the answer to the user's question and the **QA_NER method** in charge of finding the key entity (or entities) and forming the final cypher query. Regarding the answer_Retrieval method, after having previously found the template that matches our question we are now able to fetch the cypher query out of the chosen template. Afterwards, we check to see if our query has any entity placeholders that need to be replaced; if it does we call the QA_NER method. After forming the cypher query we need, then we can finally query our Knowledge Graph and return its answer.

Regarding the QA_NER method, we start by importing the Spacy library that we are going to use for NER. We apply the NER on our input question after truecasing it in order to collect all the Named Entities. After that we must check if the query has 1 placeholder or more. At this point we have to notice the fact that if our query has more than 1 placeholder and if it has the same amount of entity placeholders as the entities that are returned by the NER then we replace them in the same order that they were found in the original question.

```
Question:  What are the villages of Santorini?
Answer:
```

	Entity	Relation	Entity2
0	Santorini	ns0__containsPlace	{'ns0__name': 'Monolithos', 'uri': 'http://San...
1	Santorini	ns0__containsPlace	{'ns0__name': 'Finikia', 'uri': 'http://Santor...
2	Santorini	ns0__containsPlace	{'ns0__name': 'Mesa Gonia', 'uri': 'http://San...
3	Santorini	ns0__containsPlace	{'ns0__name': 'Vourvoulos', 'uri': 'http://San...
4	Santorini	ns0__containsPlace	{'ns0__name': 'Kamari', 'uri': 'http://Santori...

Figure 22. Question Answering System output example

Furthermore, after retrieving the answer for the original question and printing it to the user, based on the abilities of our QA system and the way that it has been built we can recommend to the user another possible question. That recommendation process derives from the fact that we have stored the template with the second highest similarity and we can use it as a recommendation. For our next

question recommendation process first we check if the previously selected template is also the template with the second highest similarity; if it isn't we move on with a similar approach as before and we use that as our template in order to retrieve the respective answer. Additionally, we must highlight the fact that in this case if the new template has any entity placeholders we use the same Named Entities as before.

```
Would you like to know more about:  What are some family friendly places?
Type YES/NO to continue: yes
Answer:
```

	Entity	Relation	Entity2_Name
0	Santorini	ns0__containsPlace	Kamari

Figure 23. Question Answering System output example (recommended next question)

Chapter 6

Experiments

In **Chapter 6**, we will be referring to the various experiments that we have successfully conducted on the **Question Answering System** and recorded in full detail for the purposes of this thesis. This chapter aims to describe the experiments that we carefully designed and carried out. In addition, we analyze the results and findings that came up after testing and evaluating our QA System.

As we have already thoroughly presented in Chapter 5, the Question Answering System that we have designed and built consists of **three main parts**, the **Question Classification** part, the **Template Matching** part and lastly the remaining part regarding the **Answer Retrieval**. The conducted experiments that we are going to be analyzing in the next few sections regard only the Question Classification part as well as the Template Matching part. We managed to evaluate each experiment by using various **evaluation metrics** (from the sklearn.metrics module) such as **Accuracy, Precision, Recall and f1-score** as well as consulting a **confusion matrix.** Nevertheless, the evaluation metric that we are going to be improving and that we will be mainly presenting is **Accuracy**.

$$Accuracy = \frac{Correct\ Predictions}{All\ Predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 24. Accuracy definition [13]

In the next three sections we will be analyzing the experiments that we have conducted; namely we tried out different preprocessing methods as well as different pre-trained models. We must mention the fact that the findings of the experiments being mentioned below will be presented with the help of comparison tables, bar charts and line graphs. In **Section 6.1** we will analyze the experiments we have conducted on the **Question Classification** algorithm. **Section 6.2** aims to present the second family of experiments that we have designed regarding the **Sentence Similarity** algorithm. Lastly, through **Section 6.3** we will be presenting our third and last experiment that also aims to compare the two Similarity methods and their combination in Template Matching.

6.1 Experiments on Question Classification

In Section 6.1 we will analyze the experiments we have conducted on the Question Classification part of our QA System. We have tested our subsystem on a small test set of 32 examples, we have created with tourism related questions about our case study, Santorini. Table 8 shows an example of our Test Set. You can find and download the complete test set (csv file) along with the python notebook containing the algorithm for these experiments here.

Text (Question)	Label
"Are there any quiet beaches in Akrotiri?"	BEACH
"Fun things to do in Santorini for couples"	ACTIVITY
"Traditional food to try in Santorini"	LOCAL_FOOD_DRINK
"Average weather in July"	CLIMATE
"Churches in Santorini"	TOURIST_ATTRACTION

Table 8. A few examples of our Test Set (Question Classification)

For this experiment we have created **two different preprocessing methods** named **Cleaning_1_1 and Cleaning_1_2** respectively. We have created these methods for the purposes of cleaning the word clusters that we have generated.

The first method, **Cleaning_1_1,** aims to iterate through the list of words and scan for words **shorter than 3 letters** such as punctuation marks or other two-letter words that offer little context to each category and then remove them. In addition, through Cleaning_1_1 we manage to remove any remaining **stopwords** that also offer little context and, at last, we manage to filter out any **null entries** that have either been generated or been caused by our earlier actions. The second method, **Cleaning_1_2,** aims to, firstly, iterate through each cluster to find the same words and then for each repeated word we check if it is its first occurrence, we move on if not we remove it.

We decided to evaluate our Question Classification subsystem by measuring its accuracy when we have applied:

1. **No Preprocessing (originally generated word clusters)**
2. **Preprocessing with only Cleaning_1_1**
3. **Preprocessing with Cleaning_1_1 and Cleaning_1_2**

Furthermore, in order to help improve our systems accuracy we have experimented with different values (**10, 15, 20, 25, 30**) for the parameter ***topn***. This specific parameter indicates the number of the top most similar words

6.1.1 Results

In subsection 6.1.1 we will be presenting and analyzing the results and findings of the experiments we have described above. **Table 9**, manages to organize the results and present them in a way that we can compare how the accuracy changes as we implement a different **preprocessing method** (vertical) or as we modify the ***topn* parameter** (horizontal).

	10	15	20	**25**	30
Αρχικα Clusters	0.56	0.59	0.66	0.69	0.66
Cleaning_1_1	0.62	0.62	0.69	0.72	0.69
Cleaning_1_1 + Cleaning_1_2	0.66	0.66	0.69	**0.75**	0.72

Table 9. Overview of the Question Classification Experiment results (Accuracy)

First and foremost, we must highlight the fact that there always seems to be an increase in the Accuracy of our system when we add the two preprocessing methods, Cleaning_1_1 and Cleaning_1_2 or even the one, Cleaning_1_1 comparing it to the low accuracy of the system with the original clusters.

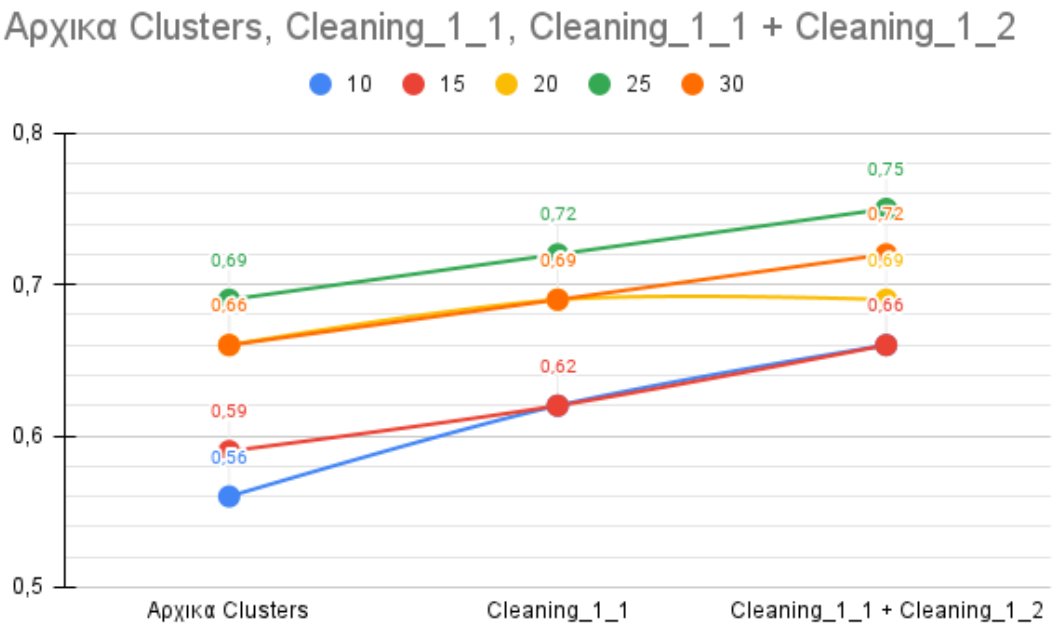

Figure 25. Line Graph showing the results of the Question Classification Experiments

Furthermore, if we study the table above we will notice that as we increase the number of the top most similar words from 10 to 25 the accuracy increases as well. Nevertheless, when we increase the ***topn*** from 25 to 30 there seems to be a decrease in the accuracy. At last we must point out that through these various experiments we came to the conclusion that the best preprocessing technique we can apply is to add both of the preprocessing methods as well as that 25 is a really good ***topn*** number for our case.

6.2 Experiments on Sentence Similarity

In Section 6.2 we will analyze the experiments we have conducted on the Sentence Similarity part of our Template Matching system. At this point we must point out the fact that for the purpose of this experiment and in order to calculate the accuracy we have created a small test set of n examples (n being the number of templates for each category). For instance, for these experiments we have borrowed the templates of the category Place. For more information about the original Templates we have created please refer to section 5.3, subsection 5.3.1. Additionally, we have successfully tested and evaluated our Sentence Similarity subsystem on the said Test Set that we also present below, in Table 10. You can find and download the complete test set (csv file) along with the python notebook containing the algorithm for these experiments here.

Text (Question)	Label
Are there any myths about santorini	T3
What are the villages of Santorini?	T1
What are the most romantic places for couples?	T7
What is the history of Santorini?	T0
What can you tell me about akrotiri?	T4

Table 10. A few examples of our Test Set (Sentence Similarity)

For this experiment as well we have created **two different preprocessing methods** named **utils_preprocess_text1 and utils_preprocess_text2** respectively. We have created these methods for the purposes of cleaning both the Test Set as well as the natural language part of the Templates we have created.

The first method, **utils_preprocess_text1,** is created specifically for the preprocessing of the natural language part of a template and manages to find and remove the template's entity placeholders that reside inside the < > brackets as well as any mentions of the word Santorini – with or without a capital s. In the meanwhile, the second method named **utils_preprocess_text2** aims to remove any punctuation marks, stopwords, other random characters and symbols and any leading or trailing spaces. Moreover, it manages to lemmatize and lowercase each word.

We decided to evaluate our **Sentence Similarity** subsystem by measuring its accuracy when we have applied:

0. **No Preprocessing (original test set and templates)**
1. **Cleaning Templates with utils_preprocess_text1**
2. **Cleaning Templates with utils_preprocess_text2**

3. **Cleaning Templates with utils_preprocess_text + utils_preprocess_text2**
4. **Cleaning Templates with utils_preprocess_text2 and Cleaning Questions with utils_preprocess_text2**
5. **Cleaning Templates with utils_preprocess_text1 + utils_preprocess_text2 and Cleaning Question with utils_preprocess_text2**

Additionally, we have experimented with different **pre-trained Sentence Transformers models** in order to help our system improve its accuracy. Below we enumerate the models we have experimented with:

1. **bert-base-nli-mean-tokens**
2. **all-mpnet-base-v2**
3. **all-distilroberta-v1**
4. **all-MiniLM-L12-v2**
5. **all-MiniLM-L6-v2**

During these experiments we tried to record each model's accuracy and study how well each model portrays each sentence (question) of our Test Set using BERT embeddings. At this point we must mention the fact that the last 4 pre-trained models (all-mpnet-base-v2, all-distilroberta-v1, all-MiniLM-L12-v2, all-MiniLM-L6-v2) were selected as they are the ones with the highest quality [46].

Whereas, the first model, **bert-base-nli-mean-tokens,** is considered deprecated and is reported to produce sentence embeddings of low quality. Nevertheless, we decided to include it in our experiments for the purposes of comparing the accuracy of our system when having a high quality model against having a low quality model.

6.2.1 Results

In subsection 6.2.1 we will try to present and analyze the results of the experiments described above. Table 11, manages to organize the results and present them in a way that we can compare how the accuracy changes as we implement a different preprocessing method (vertical) or as we modify the pre-trained model (horizontal).

	bert-base-nli-mean-tokens	all-mpnet-base-v2	all-distilroberta-v1	**all-MiniLM-L12-v2**	all-MiniLM-L6-v2
Αρχικα templates and questions	0.75	0.5	0.5	0.5	0.5
Cleaned Templates with utils_preprocess_text1	0.88	0.88	0.88	1.0	0.75
Cleaned Templates with utils_preprocess_text2	0.75	0.5	0.5	0.5	0.62
Cleaned Templates with utils_preprocess_text1 + utils_preprocess_text2	0.75	1.0	1.0	1.0	0.88
Clean Templates with utils_preprocess_text2 AND Cleaned Questions with utils_preprocess_text2	0.75	0.5	0.5	0.5	0.62
Cleaned Templates with utils_preprocess_text1 + utils_preprocess_text2 AND Cleaned Question with utils_preprocess_text2	0.88	1.0	0.88	**1.0**	1.0

Table 11. Overview of the Sentence Similarity Experiment results (Accuracy)

Firstly, regarding the different pre-trained SentenceTransformers models we can notice that for the majority of the high quality models (all-mpnet-base-v2, all-distilroberta-v1, all-MiniLM-L12-v2, all-MiniLM-L6-v2) the accuracy score presents a significant decrease when we don't apply the utils_preprocess_text1 whose main purpose and function is to remove the template placeholders and "Santorini" mentions and without it the model gets confused and ultimately maps the questions to the wrong Template.

Furthermore, it is obvious that for the most cases the high quality models do indeed return a high accuracy score compared to the reportedly deprecated model of bert-base-nli-mean-tokens. Judging from our results the best pre-trained models are admittedly the all-MiniLM-L12-v2 and all-mpnet-base-v2 and we can also see the fact that the bert-base-nli-mean-tokens model produces small accuracy scores and it is clearly deprecated compared to the others. We can see how the above statement is true by looking at the Bar Charts below.

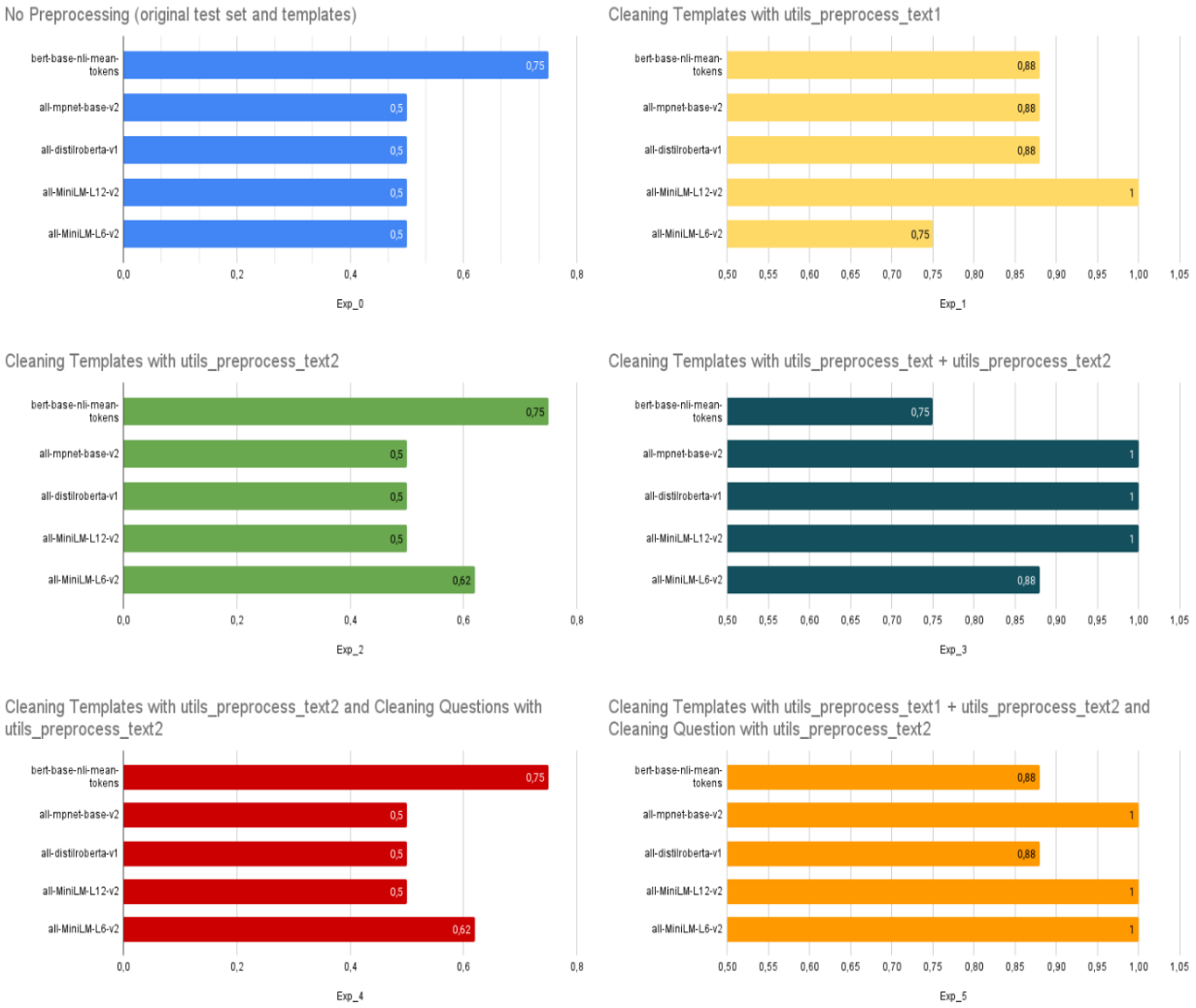

Figure 26. Bar charts showing the accuracy score of each **pre-trained model** when we apply different preprocessing methods

Moreover, regarding the different preprocessing methods we can notice that the implementation of a preprocessing method – and especially the utils_preprocess_text1 – is a really important factor to improving the subsystems accuracy. Additionally, by studying the line graph below we can see a complete picture of the results of our experiments and observe that all-MiniLM-L12-v2 manages to reach a perfect score of 1.00 when we have just preprocessed our Templates when the rest of our models reach to 0.88 at best.

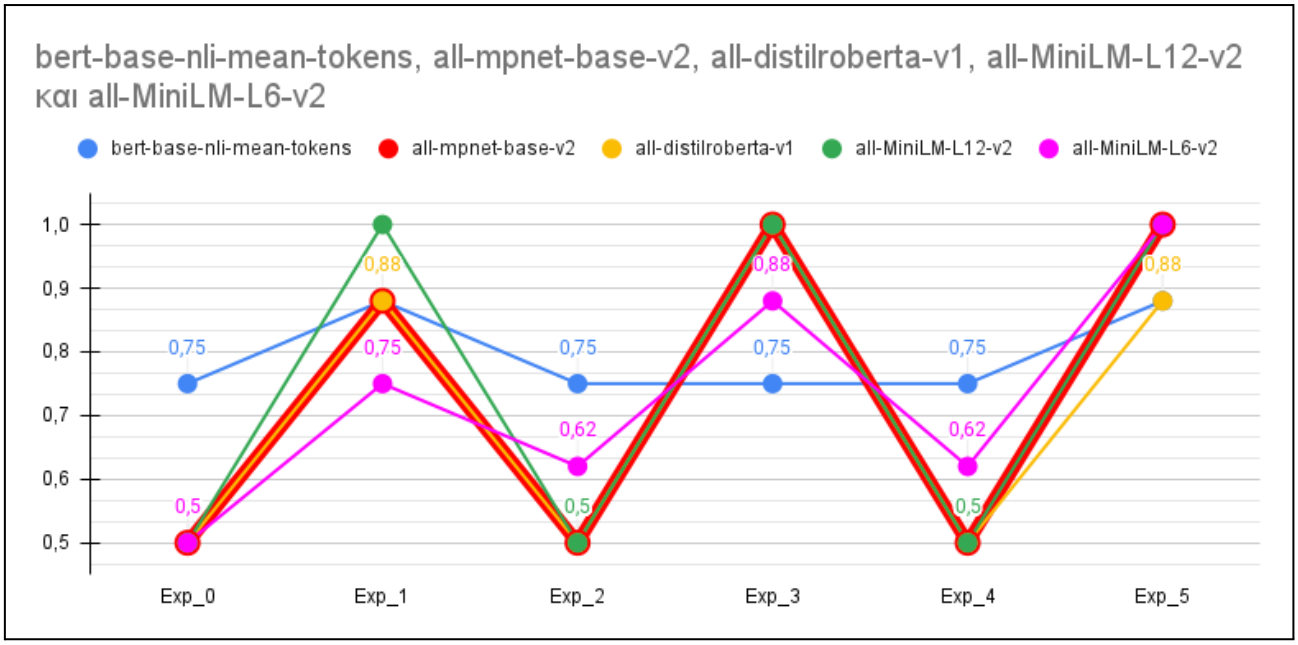

Figure 27. Line Graph showing the results of the Sentence Similarity Experiments

6.3 Experiments on Template Matching

In Section 6.3 we will analyze the experiments we have conducted on the Template Matching part of our QA System. For these experiments we have used the same Test Set mentioned in the above Section (*See Section 6.2)*. In addition, we have successfully tested and evaluated our **Template Matching subsystem** that as we have already described in detail, in Section 5.4, consists of **2 subsystems**:

- ➔ **Sentence Similarity using Sentence-Transformers**
- ➔ **Linguistic Similarity Method**

In this experiment we are going to be recording each subsystem's accuracy using their best version – that came up after conducting the two previous experiments (*See Section 6.1 and Section 6.2*) – and their worst version. After that we are going to be comparing the accuracy of each Similarity method and the accuracy of the complete (combined) Template Matching System.
For this experiment we have decided upon implementing the all-MiniLM-L12-v2 and to clean our Templates with utils_preprocess_text1 + utils_preprocess_text2 and to clean our Questions with utils_preprocess_text2 (for the best version of the Sentence Similarity part) and the same model with the original test set and templates (for the worst version of the Sentence Similarity part). Regarding the Linguistic Similarity part we have applied the same preprocessing method as above for the best version and for the worst one we have again applied no preprocessing to our templates or questions.

6.3.1 Results

In subsection 6.3.1 we will try to present and analyze the results of the experiments described above. Table 12 and Table 13, compares the different accuracy results of the best version and the worst version respectively.

	Sentence Similarity	Linguistic Similarity	Combined
Best	1.00	0.88	1.00

Table 12. Overview of the Template Matching Experiment (best version) results (Accuracy)

In the table above we can see that for the **best version** of our Template Matching subsystem both the Sentence Similarity subsystem and the Linguistic Similarity subsystem possess high accuracies of 1.00 or 0.88 thus when we combine them we get a perfect accuracy score of 1.00.

	Sentence Similarity	Linguistic Similarity	Combined
Worst	0.5	0.62	0.62

Table 13. Overview of the Template Matching Experiment (worst version) results (Accuracy)

Moreover, in Table 13 we can clearly see the importance of adding an extra, more linguistic approach to our subsystem. The worst version of our subsystem manages to produce an accuracy score of 0.5 and 0.62 for Sentence Similarity and Linguistic Similarity respectively, while the combination of those methods produces an accuracy score of 0.62. We observe that our combined accuracy doesn't deteriorate, following the Sentence Similarity accuracy but instead remains at 0.62 affected by the Linguistic Similarity.

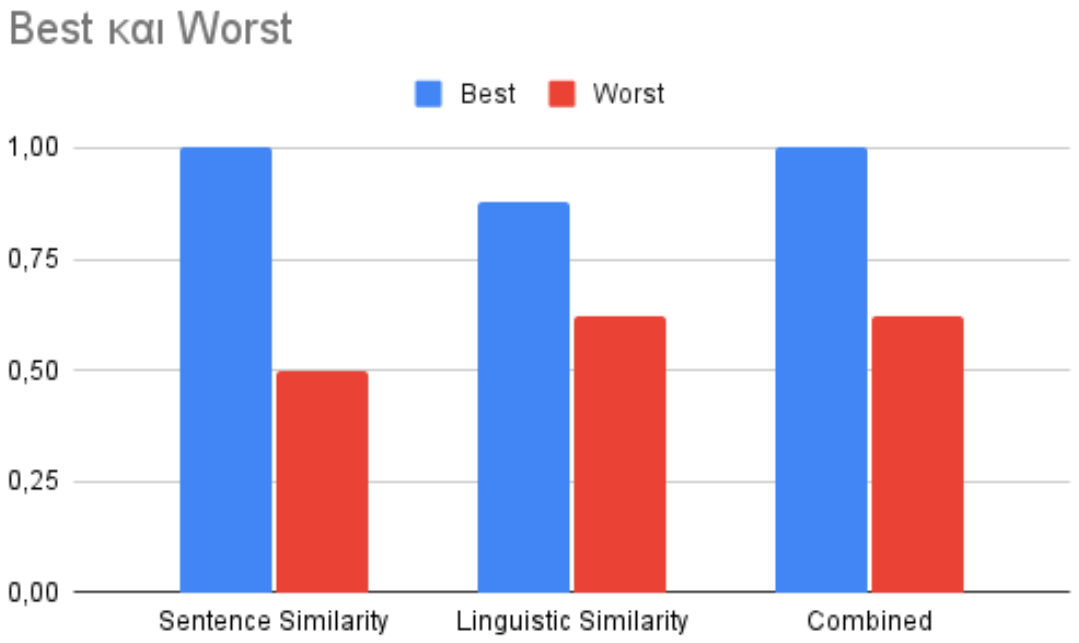

Figure 28. Bar chart showing the results of the Template Matching Experiments

Chapter 7

Conclusion

In the last chapter of this thesis, Chapter 7, we will be summarizing the previous processes and findings that have been presented and analyzed in detail in previous chapters. Additionally, we close this chapter and this study by recording our next steps and possible future work we would like to do for the ultimate improvement of our Question Answering System based on a Tourism Knowledge Graph.

7.1 Summary

In this Section we will summarize the work done for this thesis that was presented and analyzed in the previous 6 chapters. Motivated by our research problem we aimed to build a Question Answering System based on a Tourism Knowledge Graph in order to satiate the need for having a good tourist experience by being well informed and avoiding the time consuming and confusing part. In order for us to be able to develop such a system, at first, we had to get acquainted with the technologies we would be using; namely Knowledge Graphs and Question Answering Systems. We managed to familiarize ourselves with the necessary terms and definitions, the history behind them as well as any other bibliographic information we would need.

Afterwards, we moved on to focusing on the related work done on the emerging technology of Knowledge Graphs – specifically in the Tourism Domain – and how it has been implemented in different applications before. Through that we managed to gain more knowledge on the subject as well as get inspired for our own system.

After having set the background knowledge and related work foundations of Knowledge Graphs we were able to continue to its creation. We started by designing our ontology model, adding rich semantic significance to it with a well formatted ontology. We created the Tourism Knowledge Graph of Santorini that consists of 237 nodes and 285 edges following a human-included approach and using the Turtle RDF Language. We collected and cleaned mainly unstructured data and transformed them into a Knowledge Graph that was later stored in the Neo4j Graph Data Platform.

Furthermore, when the Santorinian TKG, that our QAS was going to be based on, was ready, we then moved on to the developing of our Question Answering System. The QAS we have created in a closed domain template-based QA system that consists of the Template Library, the Question Classification subsystem, the Template Matching Subsystem and lastly the Answer Retrieval subsystem. When a user submits a question to the QAS, at first the question gets classified into a

category then it is matched to a template of that category and at last according to the template's cypher query the TKG is queried and it returns the answer to the users question in the form of a table as well as a recommended follow up question that might interest the user.

We have to highlight the fact that the question classification part of our system is achieved by using BERT Embeddings and word clusters instead of a training set in order to eventually compute the cosine similarity between them and the user's question and classify the question accordingly. Moreover, regarding the Template Matching part of our system we have used two approaches; a Sentence Similarity approach where we have implemented the SentenceTransformers framework and a more Linguistic approach.

Last but not least, we have conducted various experiments on our Question Answering System. Our first experiment regards the Question Classification subsystem where we experimented with different preprocessing methods and the number of words in each word cluster and we ended up with the conclusion that the highest accuracy, ***0.75***, is achieved when we use both of our preprocessing methods and we have 25 words in each cluster. Our second experiment regards the sentence similarity subsystem where we experimented with different preprocessing methods and various SentenceTransformers pretrained models and we concluded that the best accuracy occurs when we deploy the all-MiniLM-L12-v2 model and we include the preprocessing method responsible for cleaning the natural language part of the template off of any entity placeholders and mentions of the word Santorini.

Lastly, our third experiment aims to present and compare the accuracy score of the Sentence Similarity subsystem, the Linguistic Similarity subsystem and the combination of these two subsystems. We concluded that for the best version of each subsystem, the highest accuracy score belonged to the Sentence Similarity, ***1.0***, while the Linguistic Similarity method achieved an accuracy of ***0.88***. Meanwhile, the accuracy of the combined predictions didn't get affected by the lower accuracy score of the linguistic approach and achieved a score of ***1.0***. For the worst version the highest score belonged to the Linguistic Subsystem, ***0.62***, and that affected positively the accuracy score of the combined system.

7.2 Future Work

In this Section we will present the future work that could be done to improve our system. First and foremost, regarding the Tourism Knowledge Graph we have created for the purposes of this thesis, we could further expand it by adding more knowledge – entities and relations between them – as well as integrate and maintain more dynamic data rather than just static. Moreover, regarding the Question Answering system, one thing that we would like to do in the future is try to improve our system's accuracy, either by finding ways to improve the Question Classification accuracy or by tinkering with the Linguistic approach.

Additionally, another thing that we would like to improve about our template based QAS is its Template Library and the templates that compose it. In the future a nice and important expansion of our system would be to allow more than one template to be matched to a question and of course we could always expand the template library we created by adding more templates as well. Lastly, regarding this project as a whole we would like to develop a complete system with a front and back end that will allow our system to be properly displayed and deployed by also returning the answer in the form of a graph instead of a table.

Bibliography

[1] *"11 Delicious Foods You Have to Eat in Santorini, Greece." Hand Luggage Only, 9 June 2015, handluggageonly.co.uk/2015/06/09/11-delicious-foods-you-have-to-eat-in-santorini-greece/ .*

[2] *"21 Things You Need to Know to Plan a Trip to Santorini." Santorini Secrets, 4 Feb. 2021, santorinisecrets.com/trip-to-santorini/ .*

[3] *Aasman, Jans. "Transmuting Information to Knowledge with an Enterprise Knowledge Graph." IT Professional, vol. 19, no. 6, Nov. 2017, pp. 44–51, https://doi.org/10.1109/mitp.2017.4241469. Accessed 10 Apr. 2020.*

[4] *"About: Santorini." Dbpedia.org, dbpedia.org/page/Santorini.*

[5] *Allam, A., and M. Haggag. "The Question Answering Systems : A Survey ." Www.semanticscholar.org, 2016, www.semanticscholar.org/paper/The-Question-Answering-Systems-%3A-A-Survey-.-Allam-Haggag/b2944a85b9cb428a28e30bdd236471e712667e91. Accessed 2 Feb. 2023.*

[6] *"An Introduction to Question Answering Systems." Engineering Education (EngEd) Program | Section, www.section.io/engineering-education/question-answering/. Accessed 2 Feb. 2023.*

[7] *"BERT." Huggingface.co, huggingface.co/docs/transformers/model_doc/bert.*

[8] *"Bring Rich Knowledge of People, Places, Things and Local Businesses to Your Apps." Blogs.bing.com, blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps/. Accessed 25 Jan. 2023.*

[9] *"Building the LinkedIn Knowledge Graph." Engineering.linkedin.com, engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph.*

[10] *Caballero, Michael. "A Brief Survey of Question Answering Systems." International Journal of Artificial Intelligence & Applications, vol. 12, no. 5, Sept. 2021, pp. 01–7, https://doi.org/10.5121/ijaia.2021.12501 . Accessed 15 Nov. 2021.*

[11] *Calleja, Pablo, et al. "DBtravel: A Tourism-Oriented Semantic Graph." Current Trends in*

Web Engineering, 2018, pp. 206–12, https://doi.org/10.1007/978-3-030-03056-8_19. Accessed 25 Jan. 2023.

[12] *Chang, Spencer. "Scaling Knowledge Access and Retrieval at Airbnb." Medium, 4 Apr. 2020, medium.com/airbnb-engineering/scaling-knowledge-access-and-retrieval-at-airbnb-665b6ba21e95.*

[13] *"Classification Model Evaluation Metrics in Scikit-Learn." Data Courses, 26 Feb. 2020, www.datacourses.com/classification-model-evaluation-metrics-in-scikit-learn-924/. Accessed 25 Jan. 2023.*

[14] *"Common RDF Vocabulary Labels Vocabulary." 3roundstones.com, 3roundstones.com/vocab/labels-20110606.html. Accessed 25 Jan. 2023.*

[15] *"Cracking the Code on Conversational Commerce." Www.ebayinc.com, 6 Apr. 2017, www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/. Accessed 25 Jan. 2023.*

[16] *Dadoun, A., et al. "Predicting Your next Trip: A Knowledge Graph-Based Multi-Task Learning Approach for Travel Destination Recommendation." 2021 Workshop on Recommenders in Tourism, RecTour 2021, 2021, www.semanticscholar.org/paper/Predicting-your-next-trip%3A-A-knowledge-graph-based-Dadoun-Troncy/40b7c4799ecb241044c5fbb1338a0ac84d777473. Accessed 25 Jan. 2023.*

[17] *"DBpedia." Wikipedia, 28 Nov. 2021, en.wikipedia.org/wiki/DBpedia.*

[18] *Devlin, Jacob, et al. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. 2019, arxiv.org/pdf/1810.04805.pdf.*

[19] *Diefenbach, Dennis, et al. "Core Techniques of Question Answering Systems over Knowledge Bases: A Survey." Knowledge and Information Systems, vol. 55, no. 3, Sept. 2017, pp. 529–69, https://doi.org/10.1007/s10115-017-1100-y.*

[20] *Do, Phuc, et al. "Developing a Vietnamese Tourism Question Answering System Using Knowledge Graph and Deep Learning." ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 20, no. 5, Sept. 2021, pp. 1–18, https://doi.org/10.1145/3453651. Accessed 28 Apr. 2022.*

[21] *Ehrlinger, Lisa, and Wolfram Wöß. Towards a Definition of Knowledge Graphs. International Conference on Semantic Systems (2016), ceur-ws.org/Vol-1695/paper4.pdf.*

[22] *Färber, Michael, et al. "Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO." Semantic Web, vol. 1, IOS Press, 2016, pp. 1–5,*

www.semantic-web-journal.net/system/files/swj1366.pdf.

[23] *Ferrucci, David, et al. "Building Watson: An Overview of the DeepQA Project." AI Magazine, vol. 31, no. 3, July 2010, p. 59, https://doi.org/10.1609/aimag.v31i3.2303.*

[24] *"Food Discovery with Uber Eats: Building a Query Understanding Engine." Uber Blog, 10 June 2018, www.uber.com/blog/uber-eats-query-understanding/. Accessed 25 Jan. 2023.*

[25] *"Freebase (Database)." Wikipedia, 28 Oct. 2022, en.wikipedia.org/w/index.php?title=Freebase_(database)&oldid=1118685979. Accessed 25 Jan. 2023.*

[26] *"Gartner Identifies Five Emerging Technology Trends That Will Blur the Lines between Human and Machine." Gartner, 2018, www.gartner.com/en/newsroom/press-releases/2018-08-20-gartner-identifies-five-emerging-technology-trends-that-will-blur-the-lines-between-human-and-machine.*

[27] *"Google Knowledge Graph." Wikipedia, 16 Aug. 2022, en.wikipedia.org/w/index.php?title=Google_Knowledge_Graph&oldid=1104795511. Accessed 25 Jan. 2023.*

[28] *"Google Maps." Google Maps, www.google.gr/maps/. Accessed 25 Jan. 2023.*

[29] *Greeka.com. "Greeka.com." Greeka, 2018, www.greeka.com/.*

[30] *Green, Bert F., et al. "Baseball." Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference on - IRE-AIEE-ACM '61 (Western), 1961, https://doi.org/10.1145/1460690.1460714.*

[31] *Henson, C., et al. "Using a Knowledge Graph of Scenes to Enable Search of Autonomous Driving Data." Www.semanticscholar.org, 2019, www.semanticscholar.org/paper/Using-a-Knowledge-Graph-of-Scenes-to-Enable-Search-Henson-Schmid/4e7aaa6384ed8b2caa6f42eccc7271b28def5953. Accessed 25 Jan. 2023.*

[32] *Hogan, Aidan, et al. "Knowledge Graphs." Synthesis Lectures on Data, Semantics, and Knowledge, Springer International Publishing, 2022, https://doi.org/10.1007/978-3-031-01918-0. Accessed 25 Jan. 2023.*

[33] *"Holidayify.com | Make Your Holiday Unique 2023." Holidayify.com, www.holidayify.com/. Accessed 25 Jan. 2023.*

[34] *"Hotels in Santorini, Greece | Tours, Things to Do, Travel Guides, Online Booking." Santorini-View.com, www.santorini-view.com/. Accessed 25 Jan. 2023.*

[35] *"Introducing the Knowledge Graph: Things, Not Strings." Google, 16 May 2012, blog.google/products/search/introducing-knowledge-graph-things-not/.*

[36] *Kärle, Elias, et al. "Building an Ecosystem for the Tyrolean Tourism Knowledge Graph." ArXiv:1805.05744 [Cs], July 2018, arxiv.org/abs/1805.05744. Accessed 25 Jan. 2023.*

[37] *Kärle, Elias, and Umutcan Simsek. "How to Build a Knowledge Graph." Https://2019.Semantics.cc, 9 Sept. 2019, 2019.semantics.cc/sites/2019.semantics.cc/files/how-to-build-a-knowledge-graph.pdf. Accessed 25 Jan. 2023.*

[38] *"Knowledge Graphs in End-User Products: From Cyc to AI Assistants - Part I." Www.linkedin.com, www.linkedin.com/pulse/knowledge-graphs-end-user-products-from-cyc-ai-part-daniel-kornev/. Accessed 25 Jan. 2023.*

[39] *Li, Jiahui, et al. "Towards Knowledge-Based Tourism Chinese Question Answering System." Mathematics, vol. 10, no. 4, Feb. 2022, p. 664, https://doi.org/10.3390/math10040664. Accessed 28 Apr. 2022.*

[40] *Lu, Chun, et al. "Travel Attractions Recommendation with Knowledge Graphs." Lecture Notes in Computer Science, vol. 10024, 2016, pp. 416–31, https://doi.org/10.1007/978-3-319-49004-5_27.*

[41] *"Making Search Easier." US about Amazon, 17 Aug. 2018, www.aboutamazon.com/news/innovation-at-amazon/making-search-easier.*

[42] *McComb, Dave. "Property Graphs: Training Wheels on the Way to Knowledge Graphs." Semantic Arts, 29 Oct. 2019, www.semanticarts.com/property-graphs-training-wheels-on-the-way-to-knowledge-graphs/. Accessed 25 Jan. 2023.*

[43] *Noy, Natasha, et al. "Industry-Scale Knowledge Graphs: Lessons and Challenges." Queue, vol. 17, no. 2, Apr. 2019, pp. 48–75, https://doi.org/10.1145/3329781.3332266.*

[44] *Paulheim, Heiko. "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods." Semantic Web, edited by Philipp Cimiano, vol. 8, no. 3, Dec. 2016, pp. 489–508, https://doi.org/10.3233/sw-160218.*

[45] *Pietro, Mauro Di. "BERT for Text Classification with NO Model Training." Medium, 15 Mar. 2022, towardsdatascience.com/text-classification-with-no-model-training-935fe0e42180. Accessed 31 Jan. 2023.*

[46] *"Pretrained Models — Sentence-Transformers Documentation." Www.sbert.net, www.sbert.net/docs/pretrained_models.html.*

[47] *Pundge, Ajitkumar, et al. "Question Answering System, Approaches and Techniques: A Review." International Journal of Computer Applications, vol. 141, no. 3, 2016, pp. 975–8887, www.ijcaonline.org/archives/volume141/number3/pundge-2016-ijca-909587.pdf. Accessed 2 Feb. 2023.*

[48] *"RDF 1.1 Concepts and Abstract Syntax." Www.w3.org, www.w3.org/TR/rdf11-concepts/#data-model. Accessed 25 Jan. 2023.*

[49] *"RDF 1.1 Primer." Www.w3.org, www.w3.org/TR/rdf11-primer/.*

[50] *Řehůřek, Radim, and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora." GitHub, 1 May 2010, github.com/RaRe-Technologies/gensim.*

[51] *Reimers, Nils, and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." ArXiv.org, 2019, arxiv.org/abs/1908.10084.*

[52] *"Santorini." Wikipedia, 14 Jan. 2023, en.wikipedia.org/w/index.php?title=Santorini&oldid=1133645144. Accessed 25 Jan. 2023.*

[53] *"Santorini - Wikitravel." Wikitravel.org, wikitravel.org/en/Santorini.*

[54] *"Santorini Hotels | Santorini Island Greece." Www.santorini.com, www.santorini.com/. Accessed 25 Jan. 2023.*

[55] *"Santorini Information, Traditional Products, Beaches, Volcano and Villages, Attractions, Events - Santorini Holidays 2014." Www.greektouristguides.com, www.greektouristguides.com/santorinihotelsbooking-en.html. Accessed 25 Jan. 2023.*

[56] *"Santorini.net - the Comprehensive Guide to Santorini, Greece." Https://Www.santorini.net/, www.santorini.net/.*

[57] *Schneider, E. W. "Course Modularization Applied: The Interface System and Its Implications for Sequence Control and Data Analysis." ERIC, 1973, pp. 1–21, eric.ed.gov/?id=ED088424. Accessed 25 Jan. 2023.*

[58] *"SentenceTransformers Documentation — Sentence-Transformers Documentation." Www.sbert.net, www.sbert.net/index.html.*

[59] *Sui, Yuan. "Question Answering System Based on Tourism Knowledge Graph." Journal of Physics: Conference Series, vol. 1883, no. 1, Apr. 2021, p. 012064, https://doi.org/10.1088/1742-6596/1883/1/012064. Accessed 6 Dec. 2021.*

[60] *"THE 10 BEST Santorini Wineries & Vineyards (with Photos)." Tripadvisor, www.tripadvisor.com/Attractions-g189433-Activities-c36-t132-Santorini_Cyclades_South_Aegean.html.*

[61] *Vaswani, Ashish, et al. Attention Is All You Need. 2017, proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.*

[62] *"Wikidata:Introduction - Wikidata." Www.wikidata.org, www.wikidata.org/wiki/Wikidata:Introduction.*

[63] *Wikipedia Contributors. "Cosine Similarity." Wikipedia, Wikimedia Foundation, 3 Mar. 2019, en.wikipedia.org/wiki/Cosine_similarity.*

[64] *"Question Answering." Wikipedia, Wikimedia Foundation, 12 Nov. 2019, en.wikipedia.org/wiki/Question_answering.*

[65] *Woods, W. A. "Progress in Natural Language Understanding." Proceedings of the June 4-8, 1973, National Computer Conference and Exposition on - AFIPS '73, 1973, https://doi.org/10.1145/1499586.1499695.*

[66] *WTTC. "Economic Impact | World Travel & Tourism Council (WTTC)." Wttc.org, 2021, wttc.org/Research/Economic-Impact.*

[67] *Xiao, Dinghe, et al. "A Practice of Tourism Knowledge Graph Construction Based on Heterogeneous Information." ACLWeb, Chinese Information Processing Society of China, 1 Oct. 2020, pp. 939–49, aclanthology.org/2020.ccl-1.87. Accessed 25 Jan. 2023.*

[68] *"YAGO (Database)." Wikipedia, 28 Sept. 2022, en.wikipedia.org/w/index.php?title=YAGO_(database)&oldid=1112783561. Accessed 25 Jan. 2023.*

[69] *Yang, Lu, et al. "Research on Tourism Question Answering System Based on Xi'an Tourism Knowledge Graph." Journal of Physics: Conference Series, vol. 1616, Aug. 2020, p. 012090, https://doi.org/10.1088/1742-6596/1616/1/012090. Accessed 15 Dec. 2020.*

[70] *Yochum, Phatpicha, et al. "Tourist Attraction Recommendation Based on Knowledge Graph." IFIP Advances in Information and Communication Technology, 2018, pp. 80–85, https://doi.org/10.1007/978-3-030-00828-4_9. Accessed 25 Jan. 2023.*

[71] *Zhang, Weizhen, et al. "The Chinese Knowledge Graph on Domain-Tourism." Lecture Notes in Electrical Engineering, Aug. 2019, pp. 20–27, https://doi.org/10.1007/978-981-32-9244-4_3. Accessed 25 Jan. 2023.*

[72] *Zope, Bhushan, et al. "Question Answer System: A State-of-Art Representation of Quantitative and Qualitative Analysis." Big Data and Cognitive Computing, vol. 6, no. 4, Oct. 2022, p. 109, https://doi.org/10.3390/bdcc6040109. Accessed 15 Nov. 2022.*

[73] *"ΒΑΣΙΚΑ ΜΕΓΕΘΗ ΤΟΥ ΕΛΛΗΝΙΚΟΥ ΤΟΥΡΙΣΜΟΥ." SETE, sete.gr/el/stratigiki-gia-ton-tourismo/vasika-megethi-tou-ellinikoy-tourismoy/.*

[74] *"Νοσοκομεία - Κέντρα Υγείας Σαντορίνη." Www.vrisko.gr, 2023, www.vrisko.gr/dir/nosokomeia-kentra-ygeias/santorini/.*

Appendix A.

	Natural Language Question	Cypher
Place	Historical Info about <Place>	MATCH (p) WHERE p.ns0__name = "<Place>" RETURN p.ns1__Brief_History AS Brief History;
	Villages in Santorini	MATCH (start{ns0__name:"Santorini"})-[r:ns0__containsPlace]->(end:ns1__Village) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What is Santorini also known as?	MATCH (start{ns0__name:"Santorini"})-[r:ns1__Also_Known_As]->(end) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns1__Name_Origin AS Entity2_Name_Origin
	Myths about <Place>	MATCH (start{ns0__name:"<Place>"})-[r:ns1__Myths]->(end) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__description AS Entity2_Description
	What do you know about <Place>?	MATCH p=(start{ns0__name:"<Place>"})-[r]->(end) RETURN start.ns0__name AS Entity, labels(start) AS Labels, Type(r) AS Relation, end LIMIT 10
	How can I get around in <Place>?	MATCH (start{ns0__name:"<Place>"})-[r:ns1__Get_Around]->(end) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What are some family-friendly places?	MATCH (start{ns0__name:"Santorini"})-[r:ns0__containsPlace]->(end:ns1__FamilyFriendly_Place) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What are some romantic places?	MATCH (start{ns0__name:"Santorini"})-[r:ns0__containsPlace]->(end:ns1__Romantic_Place) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
Beach	Beaches in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(end:ns0__Beach) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end AS Entity2
	Quiet Beaches in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(end:ns1__Quiet) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end AS Entity2
	Family-Friendly Beaches in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(end:ns1__FamilyFriendly_Beach) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end AS Entity2

	Natural Language Question	Cypher
	What water sports to try in <Beach>?	MATCH p=(start{ns0__name:"<Beach>"})-[r:ns1__Do]->(end:ns1__Water_Activity) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Organized beach in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(end) WHERE end:ns1__Well_Organized OR end:ns1__Partly_Organized RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Beach with blue flag in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r]->(end) WHERE end.ns1__Blue_Flag RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Where is <Beach>?	MATCH p=(start:ns1__Village)-[r]->(end:ns0__Beach) WHERE end.ns0__name="<Beach>" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What are some Quiet beaches?	MATCH p=(start)-[r:ns1__Visit]->(end:ns1__Quiet) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Where can I find sandy beaches?	MATCH p=(start:ns1__Village)-[r:ns1__Visit]->(m:ns0__Beach)-[:ns1__sandType]->(end) WHERE end.ns0__name CONTAINS 'Sand' RETURN start.ns0__name AS Entity, Type(r) AS Relation, m.ns0__name AS Entity2_Name, end.ns0__name AS Type_Of_Sand
	What are some nudist beaches?	MATCH p=(start)-[r:ns1__Visit]->(end:ns1__Nudism_Friendly) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What do you know about <Beach>?	MATCH p=(start{ns0__name:"<Beach>"})-[r]->(end) RETURN start.ns0__name AS Entity, labels(start) AS Labels, Type(r) AS Relation, end LIMIT 10
	Are there any well-organized beaches in <Place> and what can I do there?	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Visit]->(m:ns1__Well_Organized)-[r:ns1__Do]->(end) RETURN start.ns0__name AS Entity, m AS Beach , Type(r) AS Relation, end.ns0__name AS Entity2_Name

	Natural Language Question	Cypher
Activity	What can a couple do in <Place>?	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Do]->(end:ns1__Romantic_Activity) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	What to do in <Place>?	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Do]->(end:ns1__Activity) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Can I go for a <Tour> in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Do]->(end:ns1__Tour) WHERE end.ns0__name="<Tour>" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
	Activities to do in Santorini	MATCH p=(start{ns0__name:"Santorini"})-[*1..3]->(end:ns1__Activity) RETURN start.ns0__name AS Entity, end.ns0__name AS Entity2_Name
	Is <Festival> taking place during <Month>	MATCH (p) WHERE p.ns0__name="<Festival>" RETURN p.ns0__name, p.ns1__during;
	Can I try <Activity>?	MATCH p=(start{ns0__name:"Santorini"})-[*1..3]->(end:ns1__Activity) WHERE end.ns0__name="<Activity>" RETURN start.ns0__name AS Entity, end.ns0__name AS Entity2_Name
	What water sports can I do in <Beach>?	MATCH p=(start{ns0__name:"<Beach>"})-[r:ns1__Do]->(end:ns1__Water_Activity) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name
Local _Food _Drink	Famous local cuisine / products of Santorini	MATCH p=(start{ns0__name:"Santorini"})-[r]->(end) WHERE end:ns1__LocalCuisine OR end:ns1__LocalWine RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__description AS Entity2_Description
	What to eat in Santorini?	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__EatAction]->(end:ns1__LocalCuisine) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__description AS Entity2_Description
	What to drink in Santorini?	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__DrinkAction]->(end:ns1__LocalWine) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__description AS Entity2_Description

	Natural Language Question	Cypher
Notable _Person	People born in Santorini	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__birthPlace]->(end:ns1__Notable_Person) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__jobTitle AS Entity2_JobTitle
	People who died in Santorini	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__deathPlace]->(end:ns1__Notable_Person) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity2_Name, end.ns0__jobTitle AS Entity2_JobTitle
	Is <Notable Person> born in Santorini	MATCH p=(start{ns0__name:"Santorini"})-[r:ns0__birthPlace]->(end:ns1__Notable_Person) WHERE end.ns0__name="<Notable Person>" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name, end.ns0__jobTitle AS Entity_JobTitle
Climate	What is the weather of Santorini	MATCH p=(start{ns0__name:"Santorini"})-[r:ns1__weather]->(end:ns1__Climate) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end
	What is the weather like in <Month>?	MATCH p=(start{ns0__name:"Santorini"})-[r:ns1__weather]->(end:ns1__Climate) WHERE end.ns0__name CONTAINS '<Month>' RETURN start.ns0__name AS Entity, Type(r) AS Relation, end
Tourist _Attraction	What are the best attractions someone should see in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[*1..2]->(end:ns0__TouristAttraction) RETURN start.ns0__name AS Entity, end.ns0__name AS Entity_Name
	What attractions are close to <Beach>	MATCH p=(start{ns0__name:"<Beach>"})<-[r:ns1__Visit]-(middle)-[m:ns1__See]->(end:ns0__TouristAttraction) RETURN start.ns0__name AS Entity, Type(m) AS Relation, end.ns0__name AS Entity2_Name
	Churches in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[*1..2]->(end:ns1__Place_of_Worship) RETURN start.ns0__name AS Entity, end.ns0__name AS Entity_Name
	Romantic attractions for couples	MATCH p=(start{ns0__name:"Santorini"})-[*1..2]->(end:ns1__Romantic_Attraction) RETURN start.ns0__name AS Entity, end.ns0__name AS Entity_Name

	Natural Language Question	Cypher
Healthcare	Where can I find a hospital or medical center?	MATCH p=(start)-[r:ns1__healthcare]->(end) WHERE end:ns1__Medical_Center OR end:ns0__Hospital RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name
	Hospitals in <Place>	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__healthcare]->(end:ns0__Hospital) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name
Information	What should I know before going to <Place>?	MATCH p=(start{ns0__name:"<Place>"})-[r:ns1__Useful_Information]->(end)-[:ns1__answer]->(a) RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name, a.ns0__name AS Answer
	What papers do I need in order to rent a car in Greece?	MATCH p=(start:ns1__Driving_Info)-[r:ns1__answer]->(end) WHERE start.ns0__name = "Rental Car License" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name
	Where can I find an ATM?	MATCH p=(start)-[r:ns1__Useful_Information]->(end:ns1__Financial_Info) WHERE start:ns1__Village OR end.ns0__name="Cash machines (ATM's)" RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name, end.ns0__description AS Description
	What is some important driving information?	MATCH p=(start)-[r:ns1__Useful_Information]->(end:ns1__Driving_Info)-[*0..1]->() RETURN start.ns0__name AS Entity, Type(r) AS Relation, end.ns0__name AS Entity_Name, end.ns0__description AS Description, a